На правах рукописи

ЖИДЧЕНКО Виктор Викторович

ПРОГРАММНЫЙ КОМПЛЕКС МОДЕЛИРОВАНИЯ И АНАЛИЗА АЛГОРИТМОВ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

Специальность 05.13.18 –
Математическое моделирование, численные методы и комплексы программ

Автореферат
диссертации на соискание ученой степени
кандидата технических наук

Самара 2007

Работа выполнена в Самарском государственном аэрокосмическом университете имени академика С.П. Королева

Научный руководитель:

доктор технических наук, профессор Коварцев Александр Николаевич

Официальные оппоненты:

доктор технических наук, доцент Смирнов Сергей Викторович, заместитель директора по научным вопросам Института проблем управления сложными системами РАН

кандидат технических наук, доцент Попов Сергей Борисович, старший научный сотрудник Института систем обработки изображений РАН

Ведущая организация:

Центр компьютерного моделирования Нижегородского государственного университета им. Н.И. Лобачевского

Защита состоится 09.11.2007 г. в $13^{\underline{00}}$ часов на заседании диссертационного совета Д 212.215.05 в Самарском государственном аэрокосмическом университете им. академика С.П. Королева по адресу: 443086, г. Самара, Московское шоссе, д. 34

С диссертацией можно ознакомиться в библиотеке Самарского государственного аэрокосмического университета.

Автореферат разослан 08.10.2007 г.

Ученый секретарь
диссертационного совета
д.т.н., профессор А.А. Калентьев

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы.

Развитие моделей описания алгоритмов параллельных вычислений происходит с середины XX века одновременно со становлением теории последовательных вычислений.

Основными проблемами параллельных вычислений являются согласованное использование данных (синхронизация) и связанная с ней проблема тупиковых ситуаций. Для синхронизации вычислений разработано множество механизмов, таких как семафоры, критические секции, событийное управление, мониторы. Однако универсального решения, приемлемого для любых задач, до сих пор не найдено. Одни методы синхронизации не разрешают проблему возникновения тупиковых ситуаций, другие сложны для практического применения.

Параллельный вычислительный процесс приобретает особую сложность при распараллеливании на уровне подзадач и подпрограмм. В этом случае параллельные процессы могут иметь сложную внутреннюю структуру, длительность их выполнения не фиксирована; процессы взаимодействуют, обмениваются данными и обращаются к общим ресурсам. Корректная синхронизация таких процессов особенно важна для обеспечения правильности работы параллельной программы. Сложность взаимодействия параллельных процессов на уровне подзадач и подпрограмм практически исключает безошибочную работу с ними без применения средств автоматизации и делает актуальным применение методов математического моделирования для изучения взаимодействия параллельных процессов, оценки их характеристик и проверки корректности синхронизации с целью создания надежных параллельных программ.

Одной из проблем как последовательных, так и параллельных вычислений, является наглядное представление вычислительных алгоритмов с целью создания такой нотации (желательно графической), которая упрощает разработку алгоритмов конечным пользователям – специалистам в различных предметных областях.

В области теории и практики моделирования параллельных вычислений существенный вклад внесли такие ученые как С.М.Абрамов, Г. Буч, В.В. Воеводин, Вл.В. Воеводин, В.П. Гергель, Э. Дейкстра, И.Б. Задыхайло, В.Е. Котов, К. Петри, Р.Г. Стронгин, Ч.Хоар, И.Якобсон и др.

Вопросам разработки графических моделей вычислительных алгоритмов посвящены работы И.В. Вельбицкого, А.Н. Коварцева, Д.Харела и др.

Использование графических моделей позволяет ускорить разработку алгоритмов параллельных вычислений и повысить качество параллельных программ, поскольку допускает более компактное, чем текст, интуитивно понятное и легко формализуемое представление алгоритма, исключающее множество ошибок при его разработке и удобное для построения методов автоматического анализа модели.

Существует множество графических моделей описания алгоритмов параллельных вычислений, а также систем, позволяющих синтезировать программы на основе этих моделей. Каждая модель имеет специфическую область применения, в которой она обладает максимальным удобством использования и наглядностью. В

области описания вычислительных алгоритмов наиболее удобной представляется форма, близкая к традиционным блок-схемам. Проектирование вычислительных алгоритмов в модели, близкой к блок-схемам, реализовано в технологии графо-символического программирования (ГСП-технологии), разработанной на кафедре информационных систем и технологий Самарского государственного аэрокосмического университета. Модель ГСП-технологии ориентирована на последовательные вычисления, но заложенные в нее принципы позволяют перейти к описанию параллелизма. Настоящая диссертационная работа посвящена разработке графической модели алгоритмов параллельных вычислений, базирующейся на модели ГСП-технологии, а также методов и средств автоматического анализа предложенной модели, способствующих повышению качества параллельных вычислений. В результате работы создан программный комплекс, реализующий указанные методы и средства, предназначенный для построения моделей алгоритмов параллельных вычислений и автоматического синтеза параллельных программ на их основе.

Результаты работы представляют большую практическую значимость, поскольку позволяют упростить и ускорить разработку алгоритмов параллельных вычислений специалистами в различных предметных областях, предоставляя интуитивно понятную модель описания вычислений, средства автоматического анализа модели и возможность повторного использования обширного фонда алгоритмов и программ, созданных в технологии ГСП, при переходе к параллелизму вычислений.

Целью диссертационной работы является сокращение сроков разработки параллельных алгоритмов и повышение их качества за счет создания программного комплекса, предназначенного для моделирования и анализа алгоритмов параллельных вычислений с использованием наглядной графической модели и средств автоматического поиска ошибок совместного использования данных и синхронизации.

В соответствии с поставленной целью, в диссертационной работе решаются следующие ***задачи исследования***:

1) Анализ существующих подходов к моделированию параллельных алгоритмов и организации параллельных вычислений на ЭВМ;

2) Разработка графической модели алгоритмов параллельных вычислений, ориентированной на использование конечными пользователями в различных предметных областях;

3) Разработка методов автоматического поиска критических данных и анализа корректности синхронизации в алгоритмах параллельных вычислений, описываемых предлагаемой моделью;

4) Создание программного комплекса, предназначенного для построения моделей алгоритмов параллельных вычислений и их реализации на ЭВМ с параллельной архитектурой;

5) Апробация модели при решении практических вычислительных задач, а также экспериментальная проверка эффективности параллельных программ, созданных с использованием модели.

Методы исследования. В диссертационной работе используются методы математического анализа, теория графов, теория формальных грамматик, логика предикатов, численные методы.

Научная новизна. В результате проведенных исследований был получен ряд новых научных результатов:

1) Разработана новая модель описания алгоритмов параллельных вычислений, ориентированная на их наглядное графическое представление;

2) Разработаны и реализованы метод и алгоритм автоматического поиска критических данных в предложенной модели;

3) Предложен метод анализа корректности синхронизации в модели параллельного алгоритма;

4) Разработаны и реализованы метод и алгоритм поиска тупиков в алгоритмах параллельных вычислений, описываемых предложенной моделью;

5) Создан программный комплекс, содержащий визуальную среду для построения графических моделей алгоритмов параллельных вычислений, средства автоматизированного поиска критических данных и проверки корректности синхронизации параллельных вычислений, а также средства автоматического синтеза параллельных программ на основе созданных моделей.

На защиту выносятся:

- графическая модель алгоритмов параллельных вычислений;
- метод и алгоритм автоматического поиска критических данных в предложенной модели;
- метод анализа корректности синхронизации в параллельном алгоритме;
- метод и алгоритм поиска тупиков в алгоритмах параллельных вычислений, описываемых предложенной моделью.

Практическая ценность работы заключается в разработке программного комплекса моделирования алгоритмов параллельных вычислений и их реализации на ЭВМ с параллельной архитектурой. Использование программного комплекса не требует от пользователя глубоких познаний в теории параллельного программирования, что позволяет применять его для создания эффективных параллельных программ конечными пользователями, специалистами в различных предметных областях. Программный комплекс обеспечивает возможность автоматического синтеза параллельных программ на основе построенных моделей.

Реализация результатов работы. Результаты диссертационной работы нашли применение при выполнении работ по тематическому плану научно-исследовательских работ в Самарском государственном аэрокосмическом университете, финансируемых из федерального бюджета по единому заказ-наряду в 1998 г., утвержденному министерством общего и профессионального образования РФ. Результаты диссертационной работы внедрены в учебный процесс специальности 230102 – Автоматизированные системы обработки информации и управления - Самарского государственного аэрокосмического университета, что подтверждено актом внедрения.

Публикации и апробация работы. Основные положения диссертационной работы, научные и практические результаты докладывались на двух всероссийских и трех международных конференциях: IV Всероссийской научной конференции студентов и аспирантов «Техническая кибернетика, радиоэлектроника и системы управления» (Таганрог, 1998); Второй Всероссийской научной конференции «Методы

и средства обработки информации» МСО-2005 (Москва, 2005); Международном симпозиуме «Надежность и качество» (Пенза, 2002); Международном научно-практическом семинаре «Высокопроизводительные параллельные вычисления на кластерных системах» (Нижний Новгород, 2002); Первой международной конференции «Системный анализ и информационные технологии» САИТ-2005 (Переславль-Залесский, 2005).

Публикации. Всего по теме диссертации опубликовано 11 печатных работ. Список опубликованных работ приведен в заключении.

Структура и объем работы. Диссертация состоит из основной части и приложений. Основная часть содержит введение, пять глав, заключение, список использованных источников из 104 наименований. Приложение содержит тексты программ и примеры моделей. Объем диссертации – 176 страниц (без приложений), она содержит 67 рисунков и 4 таблицы.

СОДЕРЖАНИЕ РАБОТЫ

Во введении показана актуальность темы диссертации, дана общая характеристика работы, определены цели и задачи исследования. Приведены структура и краткое содержание диссертации, основные положения, выносимые на защиту.

В первой главе проводится анализ существующих моделей описания алгоритмов параллельных вычислений, а также средств и методов реализации параллельных вычислений, рассматриваются различные архитектуры параллельных вычислительных систем. Цель анализа – отражение современного уровня развития данной отрасли, основных направлений исследований, определение места диссертационной работы в общем ряду исследований, сравнение ее с аналогичными работами.

Построение эффективных параллельных вычислительных процессов невозможно без учета специфики ЭВМ, на которой будут выполняться вычисления. Рассмотрена широко применяемая классификация архитектур вычислительных систем, предложенная М. Флинном. В соответствии с ней вычислительные системы делятся на четыре класса: SISD, SIMD, MISD и MIMD. Подробно рассмотрен наиболее многочисленный класс MIMD систем, в котором выделены три подкласса: симметричные мультипроцессоры, кластеры и массово-параллельные системы. Приведены примеры существующих систем каждого класса.

Показано, что использование методов параллельной обработки информации для увеличения производительности вычислений ставит перед разработчиками новые задачи, такие как контроль совместного использования ресурсов, обнаружение и предотвращение конфликтных ситуаций, распределение вычислений между узлами параллельной вычислительной системы. Необходимость решения перечисленных задач значительно усложняет разработку алгоритмов параллельных вычислений. В главе рассмотрены два возможных способа перехода от последовательных вычислений к параллельным: автоматическое распараллеливание последовательных вычислений

(неявный параллелизм) и создание параллельных алгоритмов в явном виде (явный параллелизм). Дано описание основных результатов работ и существующих систем, реализующих оба способа.

В силу чрезвычайной сложности задачи эффективного автоматического распараллеливания произвольного вычислительного процесса, актуальными являются работы в области методов описания явного параллелизма. В главе приведено описание существующих механизмов синхронизации параллельных процессов, таких как семафоры, критические секции, синхронизация сообщениями, мониторы, описаны принципы организации асинхронных вычислений.

В результате проведенного анализа сделан вывод о перспективности разработок в области создания моделей, представляющих параллельные процессы в графическом виде. В главе рассмотрены наиболее известные графические модели параллельных процессов, такие как сети Петри, графы переходов, диаграммы потоков данных, диаграммы потоков управления, спецификация UML.

Показано, что модель описания параллельных алгоритмов должна быть наглядной, а желательно – интуитивно понятной для того, чтобы упростить работу с ней вычислителям - специалистам в различных предметных областях. Модель должна иметь четкое формальное описание, допускающее создание методов и средств ее автоматического анализа с целью обнаружения ошибок в описании параллелизма, исключения возможности возникновения тупиковых ситуаций, генерирования эффективного исполняемого кода для различных архитектур вычислительных систем.

Вторая глава посвящена описанию предлагаемой графической агрегатной модели (граф-модели) параллельных алгоритмов. Концептуальной основой для новой модели принята модель описания алгоритмов, используемая в технологии графо-символического программирования (технологии ГСП), разработанной на кафедре информационных систем и технологий Самарского государственного аэрокосмического университета.

Модель представляется четверкой <D, F, P, G>, где D – множество данных некоторой предметной области, F – множество операторов, определенных над данными предметной области, P – множество предикатов, действующих над структурами данных предметной области, $G = \{A, \Psi, \Phi, R\}$ – ориентированный помеченный граф, называемый агрегатом. $A = \{A_1, A_2, \ldots, A_n\}$ – множество вершин графа. Каждая вершина A_i помечена локальным оператором $f_i(D) \in F$. На графе задано множество дуг управления $\Psi = \{ \Psi_{1,i1}, \Psi_{1,i2}, \ldots, \Psi_{j,m}\}$ и множество дуг синхронизации $\Phi = \{\Phi_{1,i1}, \Phi_{1,i2}, \ldots, \Phi_{j,l}\}$. R – отношение над множествами вершин и дуг, определяющее способ их связи. Дуга управления, соединяющая любые две вершины A_i и A_j, имеет три метки: предикат $p_{ij}(D) \in P$, приоритет $k(\Psi_{ij}) \in N$ и тип дуги $T(\Psi_{ij}) \in N$.

Граф G инициальный – работа алгоритма начинается с выполнения оператора $f_0(D)$, помечающего начальную вершину A_0. Дальнейшее развитие вычислений ассоциируется с переходами из вершины в вершину по дугам управления. При этом переход по дуге управления возможен лишь в случае истинности предиката, которым она помечена. Если несколько предикатов, помечающих исходящие из вершины дуги,

одновременно становятся истинными, переход осуществляется по наиболее приоритетной дуге.

Тип дуги Ψ_{ij} определяется как функция $T(\Psi_{ij}) \in \{1,2,3\}$, значения которой имеют следующую семантику:

$T(\Psi_{ij}) = 1$ – *последовательная дуга* (описывает передачу управления на последовательных участках алгоритма);

$T(\Psi_{ij}) = 2$ – *параллельная дуга* (обозначает начало нового параллельного фрагмента алгоритма);

$T(\Psi_{ij}) = 3$ – *терминирующая дуга* (завершает параллельный фрагмент алгоритма).

Для описания параллелизма вводится понятие **параллельной ветви** β - подграфа графа G, начинающегося параллельной дугой и заканчивающегося терминирующей дугой. $\beta = <A_\beta, \Psi_\beta, R_\beta>$, где A_β – множество вершин ветви, Ψ_β – множество дуг управления ветви, R_β – отношение над множествами вершин и дуг ветви, определяющее способ их связи. На рис. 1 параллельные ветви обведены пунктиром.

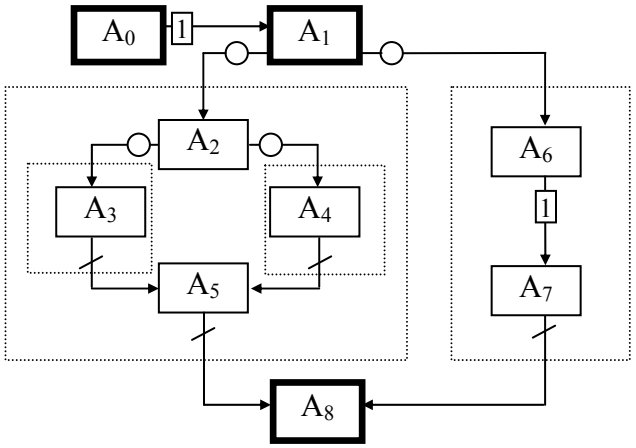


Рис. 1 – Пример граф-модели параллельного алгоритма

В соответствии с определением, каждая ветвь имеет ровно один вход и один выход, обозначаемые параллельной и терминирующей дугой.

Число параллельных ветвей в модели фиксируется при ее построении, при этом максимальное количество ветвей не ограничивается. При кодировании алгоритма, описанного с помощью предлагаемой модели, каждая параллельная ветвь порождает отдельный процесс – совокупность операторов, исполняемых последовательно на одном из процессоров параллельной вычислительной системы.

Функционирование модели начинается с запуска единственной ветви, называемой **мастер-ветвью** (мастер-процессом) β_0. Мастер-ветвь начинается с корневой вершины (вершина A_0 на рис. 1) и заканчивается в одной из конечных вершин (на рис. 1 вершины мастер-ветви имеют жирную границу). В вершинах мастер-ветви, имеющих исходящие параллельные дуги, порождаются новые параллельные ветви. Вершины этих ветвей также могут иметь параллельные дуги, таким образом, допускается вложенность параллельных ветвей.

Переход между двумя вершинами, принадлежащими различным параллельным ветвям, возможен только по параллельным и терминирующим дугам, то есть, запрещены условные переходы между вершинами различных параллельных ветвей.

Если некоторая ветвь породила новые параллельные ветви, то переход в другую вершину в ней приостанавливается до завершения работы порожденных ветвей. Таким образом, вложенные параллельные ветви исполняются последовательно относительно друг друга, а в каждый момент времени в любой ветви выполняется ровно одна вершина.

При разработке алгоритмов параллельных вычислений ключевой проблемой является синхронизация вычислений, т.е. организация их согласованного выполнения. Существуют различные способы синхронизации, такие как критические интервалы, семафоры, обмен сообщениями, мониторы. В предлагаемой модели применяется смешанный способ, использующий одновременно механизмы передачи сообщений и принципы мониторной синхронизации.

В граф-модель, содержащую вершины и дуги управления, добавляются дуги синхронизации $\Phi = \{\Phi_{1,i1}, \Phi_{1,i2}, \ldots, \Phi_{j,l}\}$. Направление дуг синхронизации определяет передачу сообщений между различными вершинами модели. Каждая дуга Φ_{ij} помечена сообщением μ_{ij}, которое отправляется вершине A_j после выполнения оператора $f_i(D)$ в вершине A_i. Сообщение в общем случае предназначено лишь для передачи информации о том, что оператор в вершине A_i завершил работу. Вершины отправляют и принимают сообщения, соответственно записывая и удаляя их из *почтового ящика*, который представляет собой список $L_{post} = [\mu_{i0,j0}, \ldots, \mu_{im,jn}]$, где $\mu_{i,j}$ – сообщение, посылаемое от вершины A_i вершине A_j.

Если в вершину A_j входит дуга синхронизации, то выполнение оператора $f_j(D)$ происходит после того, как пришло передаваемое по дуге сообщение. До этого момента граф-модель приостанавливает свою работу. В модели реализован неявный механизм передачи сообщений между вершинами одной ветви в соответствии с дугами управления, поэтому дуги синхронизации в пределах одной ветви не используются. Они применяются для определения порядка вычислений в вершинах, принадлежащих различным параллельным ветвям.

Возможна ситуация, при которой в одну вершину A_j граф-модели входит более одной дуги синхронизации. В этом случае вычисления в вершине A_j начинаются после завершения вычислений в каждой из вершин A_i, соединенных с ней дугами Φ_{ij}. Тем не менее, не всегда целесообразно ожидать завершения вычислений в каждой из вершин A_i. Например, если решается одна и та же оптимизационная задача различными численными методами, и каждый из них представлен отдельной параллельной ветвью. В этом случае, следует продолжить вычисления после получения результата в одной из ветвей, не дожидаясь окончания работы остальных.

Для предоставления такой возможности вводится понятие **семафорного предиката**. Семафорный предикат $R_{Aj} = r\,(b_{i0,j}, \ldots, b_{im,j})$, представляет собой правильно построенную формулу относительно булевых переменных $b_{ik,j}$, связанных логическими связками типа $\wedge$, $\vee$. Булевы переменные определяются следующим образом:

$$b_{kj} = \begin{cases} 1, \text{ если } \mu_{k,j} \in L_{post} \\ 0, \text{ если } \mu_{k,j} \notin L_{post}. \end{cases}$$

Семафорный предикат определяется для всех вершин, в которые входят дуги синхронизации. Для остальных вершин значение семафорного предиката принимается

тождественно истинным. Вычисление значения семафорного предиката R_{Aj} осуществляется перед запуском вершины A_j на исполнение. Если его значение ложно, то происходит циклическая проверка R_{Aj} до тех пор, пока он не станет истинным. После этого вершина A_j запускается, и выполняется оператор $f_j(D)$.

Механизм семафорных предикатов позволяет строить гибкие условия запуска вершин, определяемые видом логической формулы предиката. Использование операции «или» в семафорных предикатах повышает надежность и быстродействие программ, создаваемых с использованием модели за счет введения в нее дублирующих параллельных ветвей, работающих по различным алгоритмам на различных процессорах вычислительной системы. Если операция «или» используется в семафорном предикате вершины A_i, в которую входят терминирующие дуги нескольких параллельных ветвей, то в случае прихода сообщения от вершины одной из ветвей, остальные ветви принудительно завершаются, а управление передается вершине A_i. Благодаря этому исключается возможность нахождения модели одновременно в двух вершинах одной параллельной ветви.

В третьей главе описываются методы проверки корректности модели алгоритма параллельных вычислений. Разработанные в рамках настоящей работы методы позволяют находить в параллельном алгоритме данные, совместно используемые несколькими вершинами (**критические данные**), а также проверять корректность введенных разработчиком модели дуг синхронизации и семафорных предикатов. Корректная синхронизация подразумевает последовательное исполнение вершин, использующих критические данные, и отсутствие тупиков в вычислительном алгоритме, описываемом моделью.

Метод поиска критических данных базируется на понятии **способа использования** данных предметной области. Способ использования определяется формально как функция $H(\alpha|d)$, использующая два операнда:

α - объект граф-модели, в качестве которого может выступать отдельный оператор, подграф агрегата (ветвь) или целый агрегат;

d – данное предметной области.

Функция $H(\alpha|d)$ показывает, каким образом данное d используется объектом α.

Вводится следующая семантика для значений функции H:

$H(\alpha|d) = 0$ - d не используется объектом α;

$H(\alpha|d) = 1$ - d используется для чтения объектом α;

$H(\alpha|d) = 2$ - d используется для записи объектом α;

$H(\alpha|d) = 3$ - d – критическое данное (конфликт совместного использования данного d в объекте α)

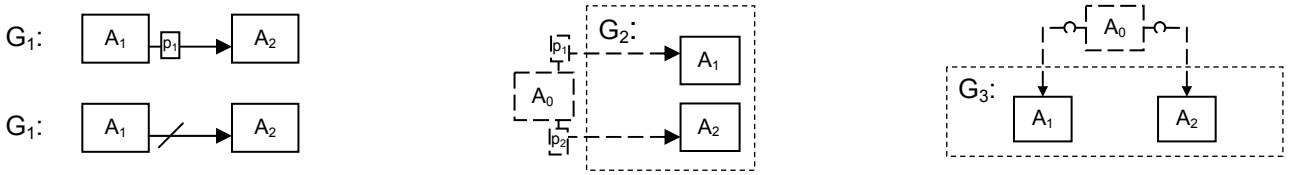
Способ использования каждого данного, используемого локальным оператором $f_i(D)$ в вершине A_i, определяется разработчиком при создании оператора. Оператор $f_i(D)$ может использовать данное либо для чтения, либо для записи. При исполнении оператора $f_i(D)$ все операции в нем выполняются последовательно. Конфликт совместного использования данных может возникнуть лишь при одновременном (параллельном) исполнении нескольких операторов $f_i(D)$, т.е. при одновременной работе нескольких вершин модели. Для этого необходимо, чтобы модель содержала параллельные ветви, и операторы $f_i(D)$ помечали вершины, принадлежащие таким

различным параллельным ветвям, которые могут исполняться одновременно. Следовательно, возможность возникновения конфликта совместного использования данных обусловлена взаимным расположением вершин, использующих эти данные.

Определение: Данное d является **критическим**, если оно используется несколькими вершинами граф-модели, которые в процессе вычислений могут исполняться одновременно. При этом хотя бы одна из этих вершин помечена объектом, использующим данное d для записи.

В работе поставлена и решена задача построения формального описания граф-модели, позволяющего обнаруживать критические данные на основе информации о взаимном расположении вершин, в которых используются эти данные.

Введем операции следования (обозначается символом Δ), ветвления (∇) и параллельного исполнения (#) над способами использования данных, определяющие способ использования данного d некоторым подграфом G_i , состоящим из вершин A_1 и A_2 граф-модели G. Указанные операции определены для трех возможных способов взаимного расположения вершин A_1 и A_2 в графе G, показанных на рис. 2.



$H(G_1|d) = H(A_1|d) \Delta H(A_2|d)$ — Операция следования

$H(G_2|d) = H(A_1|d) \nabla H(A_2|d)$ — Операция ветвления

$H(G_3|d) = H(A_1|d) \# H(A_2|d)$ — Операция паралл. исполнения

Рис. 2

Таблица истинности для введенных операций приведена в таблице 1.

Табл. 1

$H(A_1\|d)$	$H(A_2\|d)$	$H(A_1\|d) \Delta H(A_2\|d)$	$H(A_1\|d) \nabla H(A_2\|d)$	$H(A_1\|d) \# H(A_2\|d)$
0	0	0	0	0
0	1	1	1	1
0	2	2	2	2
0	3	3	3	3
1	0	1	1	1
1	1	1	1	1
1	2	2	2	3
1	3	3	3	3
2	0	2	2	2
2	1	2	2	3
2	2	2	2	3
2	3	3	3	3
3	0	3	3	3
3	1	3	3	3
3	2	3	3	3
3	3	3	3	3

Ниже приведены основные свойства введенных операций (для сокращения записи способ использования данного d в вершине A - H(A|d) – заменим обозначением самой вершины A):

1) $A\Delta B = B\Delta A$ (коммутативность операции следования)
2) $A\Delta B\Delta A = A\Delta B$ (поглощение операции следования)
3) $A\Delta (B\Delta C) = (A\Delta B)\ \Delta C$ (ассоциативность операции следования)
4) $A\Delta B = A\nabla B$ (эквивалентность операций следования и ветвления)
5) $A\#B = B\#A$ (коммутативность операции паралл. исполнения)
6) $A\#(B\#C) = (A\#B)\#C$ (ассоциативность операции паралл. исполнения)
7) $A\#A\#A = A\#A$ (поглощение операции параллельного исполнения)
8) $A\#A \neq A$
9) $A\#B\#A \neq A\#B$
10) $A\#(B\Delta C) = (A\#B)\ \Delta\ (A\#C)$, $A\#(B\nabla C) = (A\#B)\ \nabla\ (A\#C)$ (дистрибутивность операции # относительно Δ и ∇)
11) $A\Delta\ (B\#C) \neq (A\Delta B)\ \#\ (A\Delta C)$
12) $(A\#B)\ \Delta\ (A\#C)\ \Delta\ (B\#C) = A\#B\#C$

Введенные операции позволяют определить алгебру над способами использования данных в граф-модели алгоритма. Для граф-моделей с произвольной структурой, не содержащих иерархии и кратных дуг, алгебра над способами использования данных позволяет строить формулу для вычисления способа использования каждого данного произвольным объектом модели (отдельным оператором, подграфом граф-модели или целым агрегатом). Если для какого-то данного результат вычисления его способа использования агрегатом равен 3, то в рассматриваемом агрегате возможен конфликт совместного использования этого данного. Следовательно, разработчику модели необходимо позаботиться об устранении конфликта путем изменения графа модели, переименования данных или введения в модель дуг синхронизации.

В главе описан алгоритм автоматического построения и вычисления формул на основе введенной алгебры.

Наличие в модели критических данных требует от разработчика определения порядка их совместного использования с помощью механизмов синхронизации. Однако после введения дуг синхронизации и назначения семафорных предикатов могут возникнуть 2 ошибочные ситуации:

1) Введение дуг синхронизации не разрешает конфликта по данным;
2) В результате введения дуг синхронизации и семафорных предикатов возникает тупиковая ситуация, когда две вершины ожидают завершения работы друг друга (взаимная блокировка).

Для обнаружения подобных ситуаций разработаны методы, анализирующие как дуги управления, так и дуги синхронизации.

Метод проверки корректности синхронизации граф-модели проверяет, исключают ли введенные в модель дуги синхронизации конфликт совместного использования данных.

Метод применим для моделей, в которых дуги синхронизации соединяют вершины, не входящие в отличные друг от друга циклы.

Несколько изменим семантику граф-модели, добавив к каждой дуге управления параллельную ей дугу синхронизации, соединяющую те же вершины, условно положив, что при передаче управления от одной вершины к другой, передается сообщение о разрешении запуска очередного оператора.

Такое изменение позволяет привести семафорный предикат каждой вершины граф-модели к следующему универсальному виду:

$$R(A_i) = (\bigvee_j b_{ji}) \wedge (r(b_{k1,i}, \ldots, b_{kM,i})), \quad j = j1, \ldots, jn$$

где j1, …, jn – номера вершин, из которых исходят дуги управления, входящие в A_i, k1,…,kM – номера вершин, из которых исходят дуги синхронизации, входящие в A_i, а $r(b_{k1,i}, \ldots, b_{kM,i})$ – логическая функция.

Поскольку сообщения в модели предназначены для передачи информации о завершении оператора в вершине-отправителе сообщения, то можно считать, что после вычисления семафорного предиката, в момент запуска оператора вершины на исполнение, все вершины, сообщения от которых учитываются при вычислении семафорного предиката, завершили исполнение.

Принятие семафорным предикатом значения истинности в результате его вычисления можно считать условием запуска оператора вершины на исполнение. Действительно, для того, чтобы началось вычисление семафорного предиката некоторой вершины, необходимо, чтобы эта вершина получила управление по дуге управления. После этого единственным условием запуска оператора вершины на исполнение является истинность семафорного предиката данной вершины. В рекурсивном виде условие запуска оператора произвольной вершины A_i можно записать так:

$$R'(A_i) = (\bigvee_{j=j1}^{jn} (b_{ji} \wedge R'(A_j))) \wedge (r((R'(A_{k1}) \wedge b_{k1,i}), \ldots, (R'(A_{kM}) \wedge b_{kM,i}))),$$

$$R'(A_0) = \text{«истина»},$$

где j1, …, jn – номера вершин, из которых исходят дуги управления, входящие в A_i, k1,…,kM – номера вершин, из которых исходят дуги синхронизации, входящие в A_i, а $r(b_{k1,i}, \ldots, b_{kM,i})$ – логическая функция.

Введенное условие запуска оператора вершины на исполнение назовем *обобщенным семафорным предикатом***.**

В работе сформулировано и доказано следующее утверждение.

Утверждение 1: Критическое данное d, используемое вершинами $A_1, \ldots, A_n$, не приводит к конфликту тогда и только тогда, когда для любых двух вершин $A_i, A_j \in \{ A_1, \ldots, A_n \}$, обобщенный семафорный предикат одной вершины содержит сообщение от другой вершины.

Метод проверки корректности синхронизации множества вершин $A' = \{A_{i1}, \ldots, A_{in}\}$, использующих некоторое критическое данное d, заключается в построении обобщенного семафорного предиката для каждой из вершин $A_{i1}, \ldots, A_{in}$ и анализе содержимого полученных семафорных предикатов. Если существует такая

пара вершин $\{A_i, A_j\} \subset A'$, для которой семафорный предикат вершины A_i не содержит сообщения от вершины A_j, и при этом семафорный предикат вершины A_j не содержит сообщения от вершины A_i, то конфликт совместного использования данного d вершинами $A_{i1}, \ldots, A_{in}$ не разрешен.

Метод поиска тупиков (взаимных блокировок) использует понятие *состояния* модели алгоритма параллельных вычислений как множества вершин, в которых происходят вычисления в момент времени t: $S_t = \{ A_{i_0}, \ldots, A_{i_{k-1}} \}$.

В работе предложен алгоритм построения *покрывающего дерева*, определяющего множество возможных состояний модели алгоритма. Каждая вершина дерева описывает одно состояние, которое представляется множеством $S_i=\{ A_{i_0}, \ldots, A_{i_{k-1}} \}$.

Метод поиска тупиков заключается в анализе покрывающего дерева модели. Если покрывающее дерево содержит вершину без потомков (лист), множество S_i которой не равно $\{A_N\}$, где A_N – конечная вершина граф-модели, это означает, что из состояния S_i невозможен переход в другое состояние, т.е. модель содержит тупик. Вершины, в которых остановятся вычисления при возникновении этого тупика, определяются значением множества S_i.

Если все вершины покрывающего дерева, не имеющие потомков, содержат множество $S_i = \{A_N\}$, где A_N – конечная вершина граф-модели, то в модели не может возникнуть состояния, из которого не возможен переход в другое состояние, т.е. в модели нет тупиков.

В четвертой главе приводится описание программного комплекса, предназначенного для моделирования алгоритмов параллельных вычислений с использованием предлагаемой модели. В ходе работы создан программный комплекс PGRAPH 1.0, который позволяет строить граф-модель параллельного алгоритма и на основе этой модели автоматически синтезировать и запускать на исполнение параллельные программы. Структура программного комплекса приведена на рис 3.

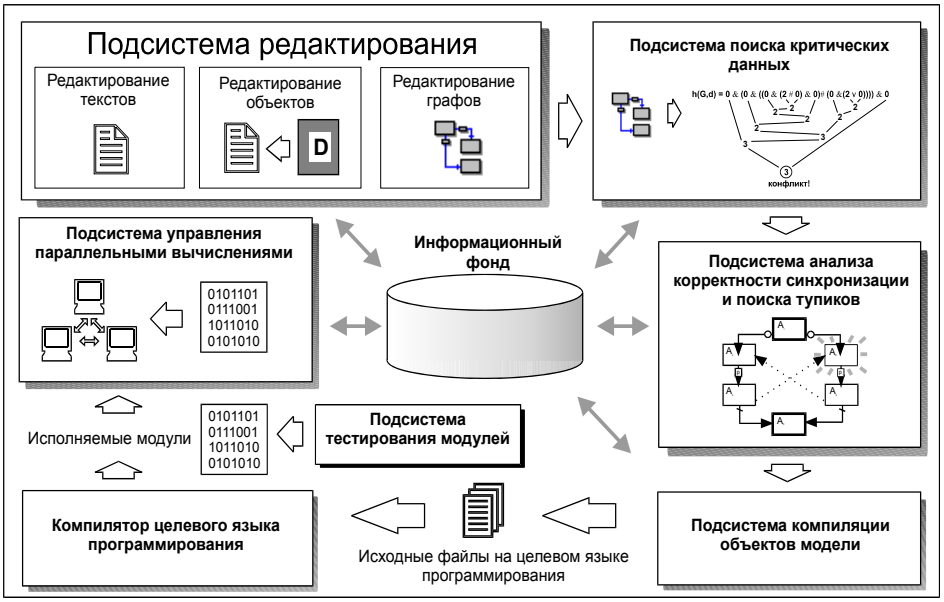


Рис. 3 – Структура программного комплекса PGRAPH 1.0

Подсистема редактирования предоставляет удобный пользовательский интерфейс для визуального построения граф-моделей, описания типов и данных предметной области, редактирования операторов, которыми помечены вершины модели, создания предикатов и средств синхронизации – сообщений и семафорных предикатов. Операторы в вершинах граф-модели представляются последовательными подпрограммами, исходный текст которых записывается на языке C++.

Программный комплекс содержит подсистемы автоматического поиска критических данных в модели параллельного алгоритма, анализа корректности синхронизации и поиска тупиков, основанные на методах и алгоритмах, разработанных в рамках данной диссертационной работы.

Подсистема компиляции позволяет на основании описания объектов модели, хранящегося в информационном фонде системы, генерировать исходные тексты на языке C++, предназначенные для компиляции в среде программирования Microsoft Visual C++ 6.0. В качестве компилятора исходных текстов используется программный продукт Microsoft Compiler, входящий в состав указанной среды программирования.

Подсистема тестирования модулей предназначена для проведения автоматизированного динамического тестирования последовательных программных модулей, созданных на основе объектов модели, для повышения надежности последовательных участков параллельного алгоритма.

Подсистема управления параллельными вычислениями осуществляет запуск скомпилированной параллельной программы на исполнение.

Программный комплекс PGRAPH 1.0 ориентирован на работу в модели передачи сообщений. Генерируемые программы могут исполняться как на ЭВМ с общей памятью, так и в распределенных системах. Механизм передачи сообщений между параллельными процессами базируется на технологии MPI (Message Passing Interface).

Программный комплекс работает под управлением операционной системы семейства Microsoft™ Windows (98, 2000). Вместе с тем, поддерживается возможность генерации параллельных программ для любой операционной системы, для которой имеется реализация MPI. Например, для создания программы, ориентированной на Linux-кластер, необходимо лишь наличие библиотеки MPI и компилятора для соответствующей версии Linux.

Информационный фонд программного комплекса хранится в базе данных, поддерживаемой СУБД MySQL.

Объем программного комплекса составляет 18.7Мб, он требует для работы не менее 32Мб оперативной памяти.

Программный комплекс PGRAPH 1.0 зарегистрирован в Отраслевом фонде алгоритмов и программ (свидетельство о регистрации №6314 от 13.06.2006 г.)

Пятая глава посвящена апробации предложенных концепций на примерах описания реальных алгоритмов параллельных вычислений.

Были построены графические модели нескольких алгоритмов решения вычислительных задач, опробованы методы поиска критических данных и проверки корректности синхронизации, экспериментально исследована эффективность параллельных программ, синтезируемых разработанным программным комплексом на основе графической модели алгоритма.

В первом эксперименте исследовался алгоритм решения линейного дифференциального уравнения в частных производных методом конечных разностей.

Рассматривалась задача Дирихле для уравнения Лапласа вида:

$$d^2u/dx^2 + d^2u/dy^2 = 0, \quad (x,y) \in D,$$

$$u(x,y)=\varphi(x,y), \quad (x,y) \in \Gamma,$$

где $D \in R^2$ – двумерная область, представляющая собой прямоугольник со сторонами, параллельными осям координат, Г – граница области D.

Алгоритмическая простота решения данной задачи, легко допускающей распараллеливание, сделала ее классическим примером в области параллельных вычислений.

В главе приводится последовательный алгоритм решения данной задачи и способ его распараллеливания, основанный на геометрической декомпозиции задачи. На рис. 4 показан способ распараллеливания задачи и графическая модель параллельного алгоритма.

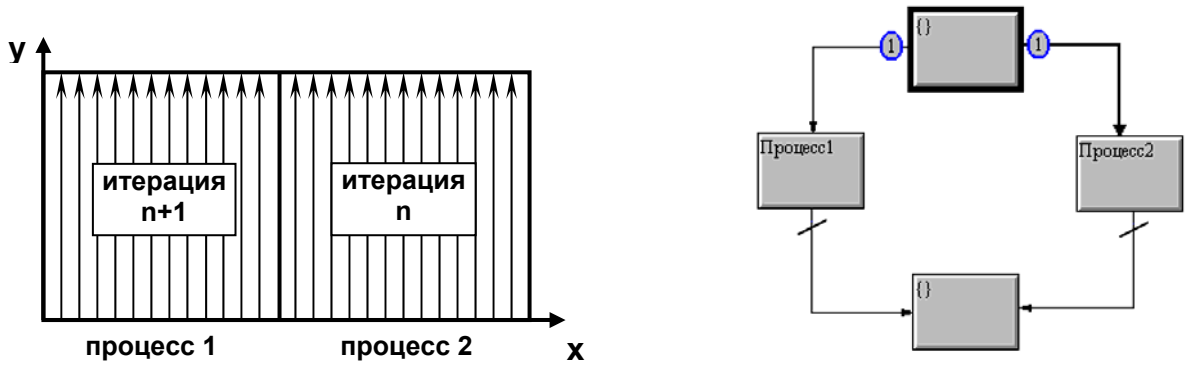


Рис. 4

Для параллельного алгоритма было изучено влияние параметров задачи и характеристик вычислительной системы на производительность реализующей его параллельной программы, а также получена оценка максимально возможного ускорения. Ускорение работы программы определяется как отношение времени работы последовательной программы к времени работы ее параллельного аналога:

$$S_{max} = T_{посл}/T_{||}$$

Показано, что для выбранного способа распараллеливания использование двухпроцессорной вычислительной системы позволяет получить двукратный выигрыш в скорости решения задачи.

Полученная оценка используется для определения эффективности программ, созданных в разработанном программном комплексе с использованием предлагаемой модели. Сравнение проводилось между результатами работы нескольких версий программы, реализующей решение уравнения Лапласа: последовательной программы, написанной на языке C++, параллельной программы на языке C++ с использованием библиотеки MPI, а также параллельной программы, сгенерированной программным комплексом на основе графической модели. На рис. 5 изображены графики зависимости времени работы каждой программы T от размерности задачи N.

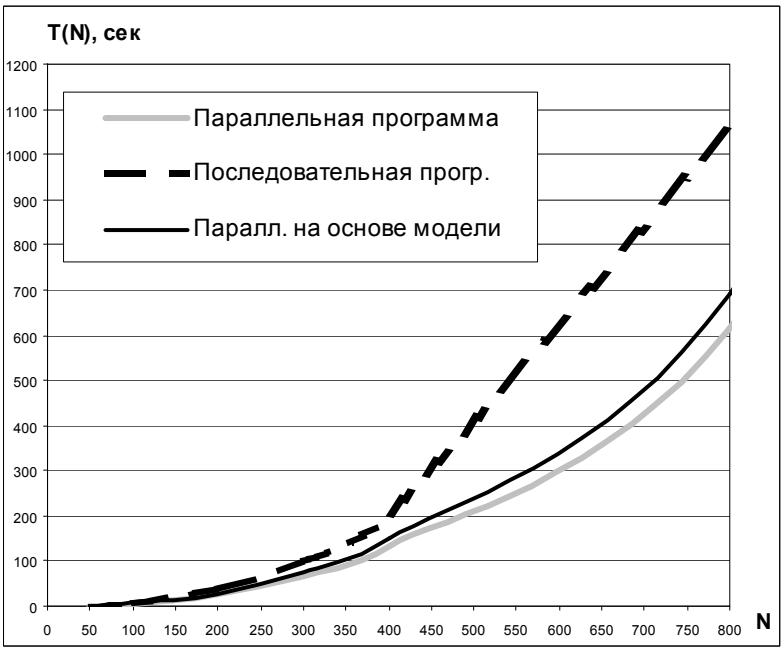


Рис. 5

Из рисунка видно, что производительность параллельной программы, автоматически сгенерированной программным комплексом PGRAPH 1.0 на основе модели параллельного алгоритма, близка к производительности аналогичной программы, написанной на текстовом языке программирования без применения средств моделирования. Разница в производительности составляет 10-12% и обусловлена применяемым методом организации вычислений на основе модели. В перспективе возможна модернизация этого метода с целью повышения производительности генерируемых программ.

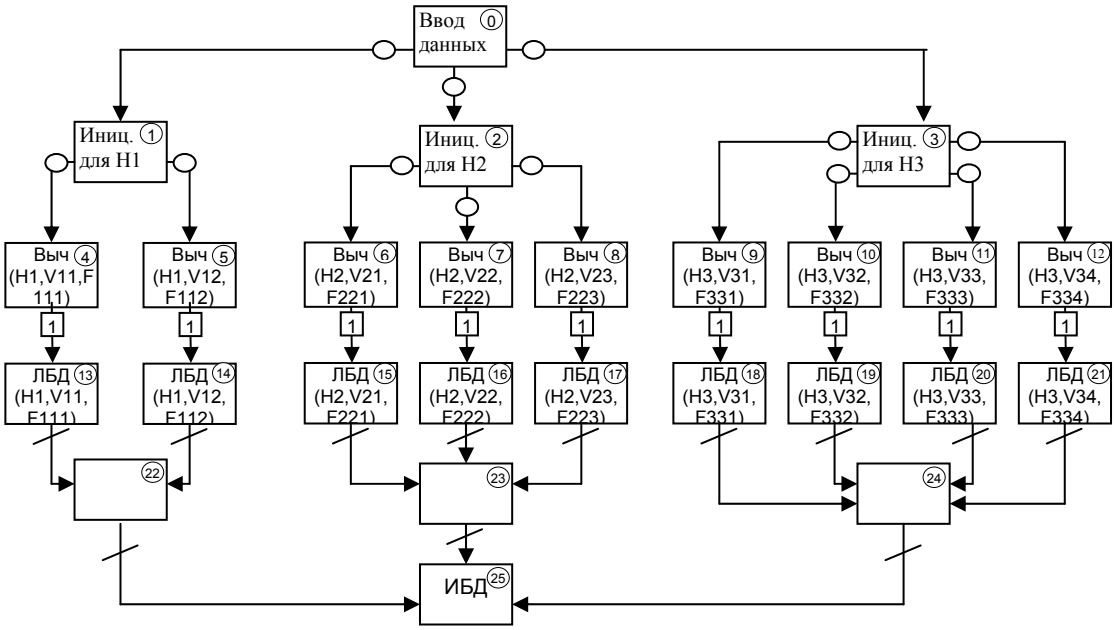
В рамках второго эксперимента была проведена проверка наглядности модели при описании вычислительных алгоритмов. Была рассмотрена задача численного решения системы дифференциальных уравнений Навье-Стокса, описывающих обтекание летательного аппарата потоком газа[1]:

$$\frac{\partial F}{\partial t} = W(F, \frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial^2 F}{\partial x^2}, \frac{\partial^2 F}{\partial y^2}, \frac{\partial^2 F}{\partial x \partial y}),$$

где $F = (F_1(t,x,y), F_2(t,x,y), \ldots, F_N(t,x,y))^T$ – вектор искомых функций.

На рис. 6 изображено графическое представление модели параллельного алгоритма, использующего принципы распараллеливания, предложенные в работе Г.А. Тарнавского и др.

[1] Вычислительная система "ПОТОК-3": опыт параллелизации программного комплекса. Часть 1. Идеология распараллеливания /Г.А. Тарнавский, В.Д. Корнеев, Д.А. Вайнер и др. // Вычислительные методы и программирование. - 2003. - Т.4, №1. - С. 37-48.



$R(A_{22}) = b_{13,22} \wedge b_{14,22}$
$R(A_{23}) = b_{15,23} \wedge b_{16,23} \wedge b_{17,23}$
$R(A_{24}) = b_{18,24} \wedge b_{19,24} \wedge b_{20,24} \wedge b_{21,24}$
$R(A_{25}) = b_{22,25} \wedge b_{23,25} \wedge b_{24,25}$

Рис. 6

Распараллеливание проводилось по двум основным определяющим параметрам задачи - высоте H и скорости V полета (т.н. глобальная параллелизация).

В главе показано, что приведенный на рис. 6 граф практически идентичен схематическому изображению структуры вычислительного процесса, которое используется для пояснения принципа распределения вычислений по процессорам вычислительной системы (схеме распараллеливания). В отличие от последней, граф-модель рис. 6 позволяет автоматически генерировать параллельную программу, реализующую вычисления. Таким образом, разработанная модель удовлетворяет поставленным требованиям: она облегчает создание параллельных программ профессионалам-вычислителям в различных предметных областях.

В третьем эксперименте проведена апробация модели при описании более сложных вычислительных алгоритмов. Рассмотрена модель многосеточного алгоритма, предназначенного для решения дифференциальных уравнений в частных производных эллиптического типа, получившего название «универсальная многосеточная технология» (УМТ)[2]. Граф-модель алгоритма, распараллеленного в соответствии с принципами, изложенными в работе С.И. Мартыненко[3], приведена на рис. 7.

---

[2] Мартыненко С.И. Универсальная многосеточная технология для численного решения дифференциальных уравнений в частных производных на структурированных сетках // Вычислительные методы и программирование. - 2000. - Т.1, №1. - С. 83-102.

[3] Мартыненко С.И. Распараллеливание универсальной многосеточной технологии // Вычислительные методы и программирование. - 2003. - Т.4, №1. - С. 49-55.

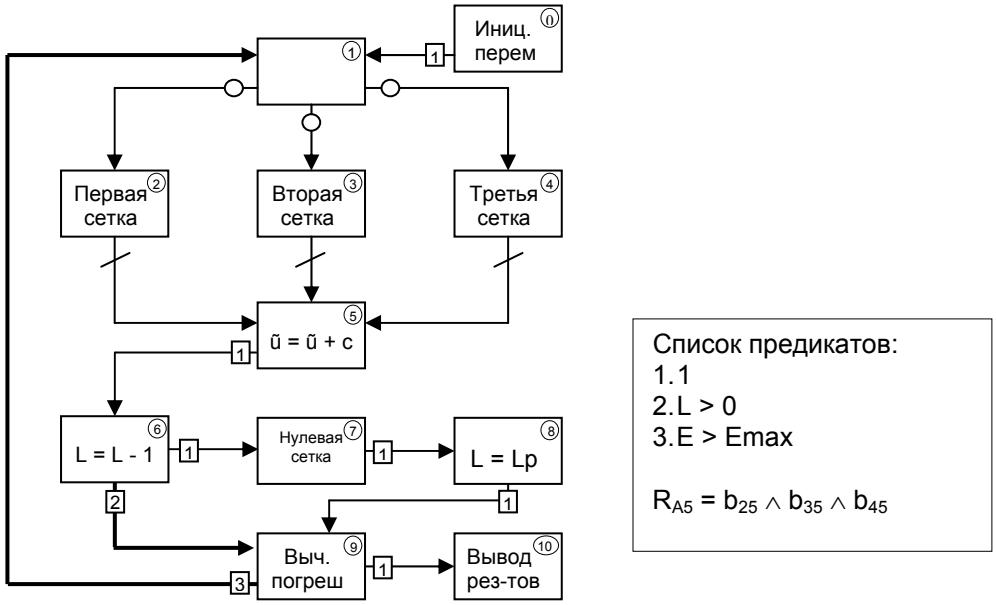


Рис. 7.

Для приведенной модели проведена апробация метода поиска критических данных - построена формула над способами использования данных (для сокращения записи способ использования данных в вершинах модели заменен номером соответствующей вершины):

$$H(G|d) = 0 \Delta (1\Delta (2\#3\#4)\Delta 5 \Delta 6 \Delta (9 \nabla 7\Delta 8\Delta 9)) \Delta 10 \qquad (1)$$

Вычисления по формуле (1), выполненные путем подстановки способа использования каждого данного в конкретной вершине, показали, что модель не содержит критических данных.

В заключении обобщаются результаты проведенного исследования, приводится список опубликованных работ.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

1) Проведен анализ существующих подходов к моделированию параллельных алгоритмов и организации параллельных вычислений на ЭВМ. По результатам анализа сделан вывод об актуальности разработок в области создания графических моделей параллельных алгоритмов, использующих явный параллелизм при описании вычислений и ориентированных на разработку программ для широко распространенных вычислительных систем с массовым параллелизмом, таких как SMP-системы на базе многопроцессорных персональных ЭВМ и кластеры.

2) Разработана модель алгоритмов параллельных вычислений, ориентированная на их наглядное графическое представление. Модель допускает автоматическую генерацию эффективных программ для широко распространенных вычислительных систем с параллельной архитектурой. Это значительно облегчает работу с моделью специалистам в различных предметных областях.

3) Разработаны методы и алгоритмы автоматического поиска критических данных, анализа корректности синхронизации и поиска тупиков в параллельном алгоритме, описываемом предлагаемой моделью.

4) Создан программный комплекс, содержащий визуальную среду для построения графических моделей алгоритмов параллельных вычислений, средства автоматизированного поиска критических данных и проверки корректности синхронизации параллельных вычислений, а также средства автоматического синтеза параллельных программ на основе созданных моделей.

5) Проведено исследование предложенной модели и разработанного программного комплекса при их использовании для моделирования и анализа вычислительных алгоритмов, допускающих распараллеливание. Показано, что предложенная модель описания алгоритмов параллельных вычислений позволяет наглядно изображать структуру алгоритма, а созданный программный комплекс сокращает сроки разработки и способствует повышению качества параллельных алгоритмов за счет средств автоматизированной проверки корректности синхронизации и поиска тупиковых ситуаций, позволяет автоматически синтезировать эффективные параллельные программы на основе графических моделей алгоритмов.

ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИОННОЙ РАБОТЫ

Публикации в изданиях, рекомендованных ВАК:

1. Жидченко В.В. Метод нахождения границ области определения алгоритмических функций в стохастическом тестировании программных продуктов. // Обозрение прикладной и промышленной математики. – 2001. – Т. 8, Выпуск 2. – С. 590.

Публикации в сборниках научных трудов:

2. Жидченко В.В., Коварцев А.Н. Метод ограничения разрядной сетки ЭВМ для автоматизированного тестирования комплексов программ // Вестник СГАУ, серия «Актуальные проблемы радиоэлектроники» / Самарский государственный аэрокосмический университет – 2000. – Вып. 4. – С. 85 - 91.

3. Жидченко В.В. Современные средства межпрограммного взаимодействия в сложных программных комплексах // Перспективные информационные технологии в научных исследованиях, проектировании и обучении. Сб. науч. тр. / Самарский государственный аэрокосмический университет – 2001. – С. 45 – 52.

4. Коварцев А.Н., Жидченко В.В. Программный комплекс моделирования параллельных вычислительных процессов PGRAPH 1.0 // Инновации в науке и образовании. - 2006. - № 6. – С. 7.

Публикации в трудах международных конференций:

5. Жидченко В.В. Повышение качества разработки параллельных программ // НАДЕЖНОСТЬ И КАЧЕСТВО. Труды международного симпозиума / Под ред. Н.К. Юркова. – Пенза: Изд-во Пенз. гос. ун-та. - 2002. – С. 134-135.

6. Жидченко В.В. Автоматизация разработки параллельных программ в технологии графо-символического программирования // Материалы второго Международного научно-практического семинара / Под ред. проф. Р.Г. Стронгина. Нижний Новгород: Изд-во Нижегородского госуниверситета, 2002. – С. 341-347.

7. Жидченко В.В., Коварцев А.Н. Моделирование синхронных параллельных вычислений при построении математических моделей сложных систем // Первая

международная конференция «Системный анализ и информационные технологии» САИТ-2005: Труды конференции. В 2т. Т.2. – М.: КомКнига, 2005. – С. 154-160.

Публикации в трудах всероссийских конференций:

8. Жидченко В.В. Метод нахождения границ области определения алгоритмически заданной функции // IV Всероссийская научная конференция студентов и аспирантов «Техническая кибернетика, радиоэлектроника и системы управления» / Тез. докл. - Таганрог, 1998. С.99-100.

9. Жидченко В.В. Комплекс программ моделирования параллельных синхронных вычислительных процессов // Методы и средства обработки информации. Труды второй Всероссийской научной конференции / Под ред. Л.Н. Королева. – М.: Издательский отдел факультета вычислительной математики и кибернетики МГУ им. М.В. Ломоносова. - 2005. – С. 465-471.

Тезисы докладов:

10. Жидченко В.В. Разработка средств интеллектуальной поддержки технологии графосимволического программирования // XXV самарская областная студенческая научная конференция / Тез. докл. – Самара, 1999. С.113-114.

11. Жидченко В.В. Подсистема автоматизированного тестирования функциональных модулей технологии графо-символического программирования // 50^{я} студенческая научно-техническая конференция / Тез. докл. – Самара, 2000. С.53.