Программный комплекс разработки алгоритмов и программ параллельных вычислений

А.Н. Коварцев, В.В. Жидченко

Самарский государственный аэрокосмический университет,
program-sistem@ssau.ru

Существенной проблемой, определяющей сложность разработки параллельных вычислительных программ, является их отладка. Это связано в первую очередь с трудностями формирования наглядного представления параллельного алгоритма и программы, особенно в текстовом виде. Кроме того, при разработке параллельных программ приходится решать такие специфические задачи, как обеспечение согласованного использования данных (синхронизация) и предотвращение тупиковых ситуаций. Ошибки в решении этих задач приводят к ошибкам в параллельной программе, которые бывает крайне сложно обнаружить в процессе тестирования.

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 [1].

Модель ГСП-технологии ориентирована на последовательные вычисления, но заложенные в нее принципы позволяют использовать ее как основу для разработки модели описания параллелизма. Настоящая работа описывает графическую модель алгоритмов параллельных вычислений, созданную на основе модели ГСП-технологии. Разработаны методы автоматического анализа предложенной модели с целью повышения качества параллельных программ, а также программный комплекс, реализующий указанные методы и 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.

Модель описания параллельных алгоритмов 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 [2]. 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.

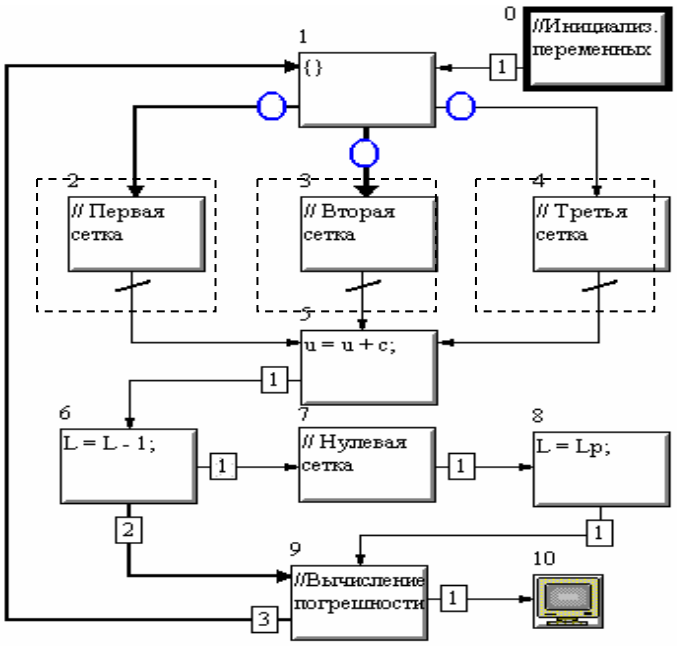


Рисунок 1. Пример граф-модели параллельного алгоритма.

В соответствии с определением, каждая ветвь имеет ровно один вход и один выход. На схеме они обозначены параллельной и терминирующей дугами.

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 Λ, V. 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.

Для уменьшения количества возможных ошибок в параллельной программе разработаны методы проверки корректности модели, которые 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;

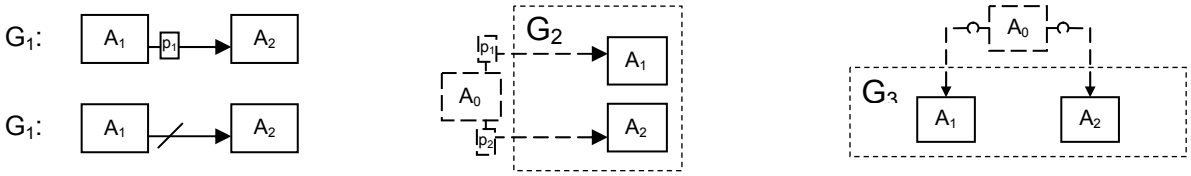
$H(\alpha|d) = 2$ - d используется для записи объектом α;

$H(\alpha|d) = 3$ - 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.

Над способами использования данных вводятся операции следования (обозначается символом Δ), ветвления (∇) и параллельного исполнения (#), 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.

Введенные операции позволяют определить алгебру над способами использования данных в граф-модели алгоритма. Для граф-моделей с произвольной структурой, не со-



$H(G_1|d) = H(A_1|d)\ \Delta\ H(A_2|d)$ $H(G_2|d) = H(A_1|d)\ \nabla\ H(A_2|d)$ $H(G_3|d) = H(A_1|d)\ \#\ H(A_2|d)$

Рисунок 2.

держащих 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.

Введение в модель алгоритма дуг синхронизации может привести к возникновению в параллельной программе взаимных блокировок (тупиков). Для предотвращения ошибок подобного рода в системе PGRAPH используется алгоритм *поиска тупиков*, который в автоматическом режиме позволяет найти все тупики программы.

Для моделирования алгоритмов параллельных вычислений с использованием предлагаемой модели и автоматического синтеза параллельных программ на их основе создан программный комплекс PGRAPH 1.0. Структура программного комплекса приведена на рис 3.

Рисунок 3. Структура программного комплекса PGRAPH 1.0.

Литература

1. Коварцев А.Н. Автоматизация разработки и тестирования программных средств. - Самарский государственный аэрокосмический университет, Самара, 1999. - 150 с.
2. 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.