Р. И. Идрисов

ВРЕМЕННАЯ РАЗВЁРТКА ВНУТРЕННЕГО ПРЕДСТАВЛЕНИЯ IR2 ЯЗЫКА SISAL 3.1*

На сегодняшний день увеличение вычислительных мощностей связано уже не с ускорением отдельного, а с добавлением дополнительных вычислителей, созданием различного рода суперкомпьютеров и кластеров; в связи с этим, распараллеливание программ становится более актуальным. Для человека, решающего конкретную вычислительную задачу, удобнее всего не вдаваться в конкретные детали распараллеливания своей задачи и иметь платформенно-независимый код. Одним из возможных решений является использование специализированных языков, которые легко распараллеливаются.

Язык Sisal [1] реализует потоковую модель вычислений и является одним из самых известных языков такого типа. Он также позиционируется как замена языка Fortran для вычислений, поскольку в отличие от других потоковых языков имеет синтаксис, более схожий с привычными языками программирования, такими как Pascal. Потоковая организация вычислений обеспечивает более естественное распараллеливание кода. Механизм однократного присваивания сильно упрощает анализ зависимостей.

Компилятор языка, разрабатываемый в Институте систем информатики им. А. П. Ершова (ИСИ СО РАН), имеет только последовательную реализацию и не имеет средств к оптимизации и выявлению параллелизма. В связи с этим задача распараллеливания оптимизации Sisal является актуальной.

Целью данной работы является формулирование задачи распараллеливания в терминах внутреннего представления языка Sisal IR2.

ВНУТРЕННЕЕ ПРЕДСТАВЛЕНИЕ

Для исследования параллельных свойств программ, записанных на императивных языках, используются **графы зависимостей** [2]. Графом зависимостей называется граф, построенный для некоторой программы, в котором вершины соответствуют её операторам, а дуги соединяют две вершины

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 07-07-12050).

в случае, если соответствующие операторы используют одну переменную. Минимальным снизу графом зависимостей называется граф, в котором вершина имеет только по одной входящей дуге для каждой из используемых переменных, и эта дуга идёт из ближайшей предшествующей по исполнению к ней вершины (так как программа последовательная, и порядок чётко определён, ближайшая вершина — единственная). Для каждого из четырёх видов зависимостей можно построить такой граф. Мы будем рассматривать императивные программы только с однократным присваиванием, поэтому нас интересует исключительно граф истинных зависимостей. Далее термин *граф зависимостей* будет обозначать минимальный снизу граф истинных зависимостей.

В языке Sisal на данный момент используется внутреннее представление IR2 [3], которое не содержит указания на последовательность кода и не зависит от архитектуры целевого компьютера; оно представляет собой иерархический граф, в котором каждая вершина означает функцию, а ребро — передачу информации. Граф реализован с помощью двух сущностей: вершина и порт. Порт отвечает за входные и выходные данные и связан с другим портом — поставщиком или получателем данных. Вершина может содержать входные и выходные порты и также подграфы, относящиеся к внутренним вычислениям, связь по данным с которыми также осуществляется посредством портов. Вершина может не содержать входных портов; такая вершина отвечает за константное значение. Сущности дуги как таковой нет, но можно считать, что дуга соединяет порты, между которыми происходит передача данных. Нашей задачей является формулировка условий задачи распараллеливания в терминах IR2 и отображение архитектурных особенностей на условия в терминах IR2.

ПОСТРОЕНИЕ РАЗВЁРТКИ

Для последовательных программ на императивных языках, представленных в виде минимального сверху графа зависимостей по данным G, временной строгой развёрткой называется вещественный функционал f(G), который возрастает вдоль дуг графа. Это означает, что если из вершины u идёт дуга в вершину v, то $f(u) < f(v)$. Для учёта реальных условий добавляются следующие векторы: h_i — вектор реализации, отвечающий за скорость выполнения операций в вершинах графа, w_{ij} — вектор задержек, отвечающий за задержки при передаче данных между вершинами и вектор s_i — вектор граничных значений, определённый для входных вершин, отвечающий

за подачу данных. Все значения ≥0, индексы i, j могут принимать значения от 1 до N, где N — число вершин графа. Временная развёртка может быть представлена набором чисел t_i, где $t_i = s_i$ для входных вершин, а для остальных $t_i >= \max(t_j + w_{ij}) + h_i$, где j принимает значения соответствующе смежным вершинам.

Временной развёрткой для внутреннего представления языка Sisal будем называть вектор t_i, где $i \in [1..N]$ и N — количество портов внутреннего представления, удовлетворяющий следующим условиям:

1. если данные передаются из порта p_i в порт p_k, то $t_i \leq t_k$,
2. если p_i — входной, а p_k — выходной порт одной вершины, то $t_i \leq t_k$.

Аналогичным образом введём вектор граничных значений s_i, который определяет значения t_i для входных портов, не связанных с другими портами, вектор задержек w_{ij} и вектор реализации h_i. В случае графа зависимости смысл векторов w_{ij} и h_i прозрачен; требуется установить их смысл для представления IR2, поскольку условия, накладываемые архитектурой, удобней формулировать в терминах выполнения конкретных операций и задержек на пересылку.

Для представления, которое содержит только простые вершины (не содержащие подграфов), построим соответствующий граф зависимостей следующим образом: каждому входному порту будет соответствовать входная вершина графа зависимостей, каждой вершине — вершина в новом графе, дуги соединяют вершины в случае, если в исходном графе соответствующие вершины были соединены через порты. Если найдена временная развёртка графа зависимости по данным, удовлетворяющая ограничительным векторам, из неё можно получить временную развёртку исходного внутреннего представления IR2 следующим образом: для входных портов компоненты вектора t примем равными значениям развёртки в вершинах графа зависимостей, для выходных портов вершины t примем равным $t_k + h_k$, где k — номер соответствующей вершины. Времена реализации этих развёрток $\max(t_i)$ будут совпадать. Таким образом, в этом случае можно сказать, что вектор реализации h_i, определённый для вершин IR2 обозначает скорость выполнения операции, расположенной в вершине, а вектор w_{ij}, определённый только для соседних портов — задержки на пересылку данных.

Для каждой вершины, содержащей подграф, можно построить граф зависимости, поскольку возможна её трансляция в последовательную программу на императивном языке, а для такой программы граф зависимостей может быть построен. Вопрос возникает для задержек wij, которые относятся к портам вершины, соединяющих внутреннюю часть с внешней. Для

таких портов будем считать внутреннюю часть пересылки фиктивной и её задержку равной 0, поскольку нет смысла осуществлять двойную пересылку данных в данной модели. Вектор реализации для составных вершин не относится напрямую к архитектурным особенностям целевого вычислителя, а скорее к особенностям подграфа, который содержится в составной вершине. Особенностью реализации представления IR2 является также то, что внутренние подграфы вершин не всегда связаны портами со своей (родительской) вершиной, хотя такая связь предполагается. Например, вершина, соответствующая циклу, содержит четыре подграфа: граф инвариантов цикла, граф тела цикла, граф постусловия и граф генерации выходного значения. Граф постусловия получает на вход кроме значений, сгенерированных в графе инвариантов, также значения, сгенерированные на предыдущей итерации и на текущей. Фактически он должен быть вычислен после двух итераций цикла (по готовности обоих наборов значений), но вторая итерация не может быть вычислена до срабатывания постусловия. Первая итерация не может быть вычислена до срабатывания предыдущей, потому что связана с ней по данным. Из этих примеров видно, что нельзя просто соединить внутренние порты составных вершин и провести анализ практически так же, как и для графа зависимостей. Для составной вершины операции выбора вычисление графа, относящегося к условию, может быть вычислено до готовности остальных операндов, аналогично для других подграфов, составляющих вершину этой операции. Следовательно, составная вершина графа представления IR2 не может быть рассмотрена как простая (в виде макро-операции) без ограничения параллелизма. Это означает, что без изменения структуры задача не сводится к вычислению развёрток графов, составляющих представление.

Алгоритм построения развёрток, используемый для графов зависимостей, может быть использован при анализе графов IR2, которые не содержат подграфов. В этом случае начальные условия для нахождения временной развёртки можно обозначить также тремя векторами s, h и w, для которых s_i определено для входных портов и означает времена поступления начальных данных, h_i определено для вершин и означает скорость выполнения операций, w_{ij} определено для соседних портов и означает пересылку данных. Для «сшивки» развёрток на входе и выходе составной вершины будем пользоваться дополнительными правилами, учитывающими особенности представления IR2.

Задача нахождения временной развёртки программы, записанной в представлении IR2, сводится к нахождению вектора t, определённого для всех портов и означающего моменты времени готовности операнда порта.

Критерием качества распараллеливания естественно считать число $\max(t_i)$, которое означает время готовности последнего операнда.

Рассмотрим граф зависимостей некоторой программы G, состоящий как минимум из двух вершин. Разобьём множество его вершин V на два непустых множества V’ и V’’ произвольным образом. Построим графы G’ и G’’ таким образом, что G’ будет включать вершины из V’ и дуги, их соединяющие, а G’’ вершины из V’’ и дуги из вершин V’’ в вершины V’’. Для каждой дуги v_i->v_j из V’ в V’’ добавим выходную вершину, соединённую с вершиной v_i, в граф G’ и входную, соединённую с вершиной v_j, в граф G’’. Аналогично для дуг из V’’ в V’ добавим соответствующие входные и выходные вершины в графы G’ и G’’. Пусть ограничения заданы для графа G векторами s, h и w описанными выше.

Утверждение. Минимальная временная развёртка графа G может быть построена из минимальных временных развёрток для графов G’ и G’’ тогда, когда дуги, соединяющие вершины множеств V’ и V’’ в исходном графе G, имеют одну направленность (из V’ в V’’ или наоборот) или отсутствуют.

Доказательство. предположим, что дуги идут из V’ в V’’. Будем строить минимальную временную развёртку для G’. Она может быть построена, если доопределить векторы ограничений s, h и w для новых дуг и вершин, отсутствующих в G. Для добавленных выходных вершин примем $h_i = 0$, для добавленных дуг w_{ij}’= w_{ij}, где индексы i, j соответствуют вершинам концов дуги v_i -> v_j из V’ в V’’. Вектор граничных значений s дополнять не требуется, поскольку дополнительных входных вершин добавлено не было. Для построения развёртки графа G’’ нам также требуется дополнить векторы ограничений. Примем $h_i = 0$ для новых вершин и $w_{ij} = 0$ для новых дуг. Для каждой дуги v_i -> v_j из V’ в V’’ в граф G’’ была добавлена входная вершина. Примем значение вектора s_k для этой вершины равным значению временной развёртки в выходной вершине графа G’, которая была добавлена для этого ребра. Векторы ограничений дополнены, и развёртка может быть найдена. По построению все вершины графа G содержатся в G’ либо в G’’, временную развёртку для вершин графа G примем равной найденным значениям соответствующих развёрток в графах G’ и G’’. Для того, чтобы этот вектор был развёрткой графа G, необходимо выполнение двух условий: $t_i = s_i$ для входных вершин и $t_i >= \max(t_j + w_{ij}) + h_i$ (1) — для всех остальных. Первое условие выполняется по построению, второе условие для вершин, которые не имели дуг, связывающих V’ и V’’, также выполняется по построению. Остаётся проверить для всех вершин, соединённых дугой v_i -> v_j где вершина v_i принадлежит к V’, а v_j — к V’’. Для вершин v_i проверки не

требуется, поскольку в сумме участвуют только инцидентные вершины, а они никак не изменились. Для v_j по построению значение вектора развёртки будет $t_j >= \max(s_k + 0) + 0$ (2), где 0 подставлены вместо добавленных w_{ij} и h_k новых вершин и рёбер, которые мы приняли равными 0 для графа G'', а s_k-значения вектора ограничений для новых вершин. Этот вектор может быть расписан через значение развёртки в графе G' как $s_k >= \max(t_l + w_{kl}) + 0$, поскольку в каждую добавленную выходную вершину графа ведёт только одна дуга $s_k >= t_l + w_{kl}$. Из минимальности развёртки следует, что $s_k = t_l + w_{kl}$, так как значение w для новых рёбер соответствовало значению для ребра из V' в V''. При подстановке в неравенство условия (2) получим условие для развёртки (1). Верно и обратное: если развёртка графа G удовлетворяет условию (1), то она порождает развёртку для графов G' и G'' если принять вектор начальных условий s_k для добавленных вершин равным $t_l + w_{kl}$; в противном случае развёртка не будет минимальной. Таким образом, если построенная развёртка G не является минимальной, значит, есть такая вершина, для которой значение t может быть уменьшено (по определению минимальная развёртка минимальна для всех вершин графа), значит развёртка какого-то из графов G' или G'' может быть уменьшена, чего не может быть, поскольку они минимальны. Получаем, что развёртка графа G, построенная таким образом, является минимальной. Для случая, когда дуги идут из V'' в V', доказательство аналогично, в случае отсутствия дуг — тривиально.

Для построения минимальной развёртки графа G через вычисление развёрток для графов G' и G'' в случае, если в исходном графе существует дуга из V' в V'' и существует дуга из V'' в V', потребуется неоднократное вычисление минимальных развёрток графов G' и G''. Аналогичным образом добавим в графы дополнительные вершины. Развёртка графа G' может быть вычислена только для вершин, которые не связаны по пути с вершиной-источником, заменяющей ребро из графа G''. Для графа G'' аналогично. Последовательным вычислением развёрток мы можем дополнять векторы граничных значений графов G' и G'' пока полная развёртка не будет найдена. Нахождение развёртки таким способом не завершится, если в графе G присутствовал контур, и часть его оказалась в G', а другая в G''. Но мы рассматриваем построение развёрток только для бесконтурных графов. Количество итераций не будет превосходить $\min(n_f, n_b) + 1$, где n_f,n_b количество прямых и обратных дуг соединяющих вершины V' с V'' в исходном графе G. Построенная таким образом развёртка будет развёрткой для графа G, способ и доказательство аналогично случаю для одного типа рёбер. На каждом шаге вычисления будут производиться не для всего графа G' или

G’’, а только для тех вершин, которые достижимы из вершины-источника, начальное условие для которой было определено на предыдущей итерации.

ЗАКЛЮЧЕНИЕ

Для программы, записанной в терминах внутреннего представления IR2, задача вычисления временной развёртки портов может быть сведена к задаче отдельного вычисления развёрток для его подграфов. Потребуется представление составных вершин представления графа в виде графа зависимости. Такая развёртка будет минимальной. Существенно то, что нам не обязательно приводить всю Sisal программу к виду графа зависимости, а можно ограничиться только определением структур для составных вершин. Решение задачи для портов в случае простых вершин, не содержащих подграфы, аналогично решению задачи для вершин графа зависимости.

Решение задачи о минимальной временной развёртке позволяет определить минимальное время исполнения программы на граф-машине или в условиях неограниченного параллелизма. Это время даёт оценку снизу на время исполнения программы; если значение получается неприемлемо большим — требуется изменение программы для того, чтобы компилятор смог выделить больше параллельных участков кода. Сравнение значения для временной развёртки до и после оптимизационных преобразований может служить критерием их эффективности для данной программы.

СПИСОК ЛИТЕРАТУРЫ

1. Касьянов В. Н., Бирюкова Ю. В., Евстигнеев В. А. Функциональный язык Sisal // Поддержка супервычислений и интернет-ориентированные технологии. — Новосибирск: ИСИ СО РАН, 2001. — С. 54–67.
2. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. — СПб.: БХВ-Петербург, 2002. — 608 с.
3. Стасенко А. П. Внутреннее представление системы функционального программирования Sisal 3.0 — Новосибирск, 2004. — 54 с. — (Препр. / РАН. Сиб. Отд-ние. ИСИ; № 110).