АЛГОРИТМЫ МИНИМИЗАЦИИ АППАРАТНЫХ РЕСУРСОВ ВС

Костенко В.А., Романов В.Г., Смелянский Р.Л.

Московский Государственный Университет им. М.В.Ломоносова, факультет Вычислительной математики и кибернетики, лаборатория Вычислительных комплексов
119899, Москва, ГСП-3, Воробьевы горы, МГУ, 2-й учебный корпус, ф-т ВМиК,тел.: (095) 939-46-71, факс: (095) 939-25-96, Email: kost@cs.msu.su

ABSTRACT

Investigation of efficient algorithms for the important subslass of computer architecture synthesis problems - minimization of hardware resources - is considered. Results of investigation of such algorithms are presented in this paper.

ВВЕДЕНИЕ

Целью данной работы является исследование эффективности алгоритма, описанного в работе [2]. В этой работе был предложен алгоритм для решения практически важного подкласса задач синтеза архитектур - минимизации числа процессоров вычислительных систем (ВС), в следующей постановке: для заданных поведения программы и директивного срока ее выполнения, требуется выбрать минимально достаточное число процессов в ВС и получить расписание выполнения программы. Поведение программы задается в инвариантной форме относительно архитектуры ВС [1].

ФОРМАЛЬНАЯ ПОСТАНОВКА ЗАДАЧИ СИНТЕЗА АРХИТЕКТУР ВС ОПТИМАЛЬНЫХ ПО ЧИСЛУ ПРОЦЕССОРОВ

Задача синтеза архитектуры ВС оптимальной по числу процессоров может быть сформулирована следующим образом:
Для заданных:

- $H(PR) = (P, \prec)$ - графа истории поведения программы,
- T^{dir} - директивного срока выполнения,

требуется определить:

- M - число процессоров в ВС,
- $HP = (\{SP_i\}_{i=(1...M)}, \prec_c)$ - расписание выполнения программы, в форме строго упорядоченных списков рабочих интервалов $\{SP_i\}_{i=(1...M)}$ и набора секущих ребер $\prec_c$,

при этом должны выполняться условия:

1. $T^W \le T^{dir}$ - время выполнения программы не должно превышать директивный срок,
2. $M \to min$ - число процессоров должно быть минимально необходимым для выполнения условия 1,
3. Частичный порядок, заданный H, не должен быть нарушен в HP.

Модель поведения программы задается историей выполнения программы H и набором ограничений (в рассматриваемом варианте - директивный срок выполнения программы). H представляет собой ациклический ориентированный размеченный граф: $H = (P, \prec)$. Вершинам $P = \{p_i\}_{i=1}^{N_1} \cup \{p_i\}_{i=1}^{N_2} \cup \ldots \cup \{p_i\}_{i=1}^{N_K}$ соответствуют рабочие интервалы процессов [2], дугам $\prec = \{\prec_{ik} = (p_i, p_k)\}_{(i,k) \in (1...N)}$ - связи, определяющие взаимодействия между рабочими интервалами из множества P. Где N_i - число рабочих интервалов в процессе p_i, K - число процессов в программе PR, $N = N_1 + N_2 + \ldots + N_K$ - мощность множества P. Рабочие интервалы процесса определяются точками взаимодействия процесса с другими процессами. Все рабочие интервалы одного процесса должны быть назначены на один и тот же процессор.

Для каждого процесса определен строгий порядок на множестве его рабочих интервалов. Чередование рабочих интервалов различных процессов, назначенных на один и тот же процессор, допустимо, если не нарушается частичный порядок, заданный $\prec$. Отношение $\prec_{ik}$ представляется следующим образом: если $p_i \prec_{ik} p_k$, то рабочий интервал p_i необходимо выполнить до начала выполнения рабочего интервала p_k. На частичный порядок $\prec$ накладываются условия:
1) ацикличность;
2) транзитивность (отсутствие избыточных связей, не отражающих непосредственную зависимость по данным и управлению между рабочими интервалами).
Каждая вершина имеет свой уникальный номер и метки: принадлежности рабочего интервала к процессу и вычислительной сложности рабочего интервала. Каждая дуга имеет метку обьема передаваемых данных. Вычислительная сложность рабочего интервала и объем данных обмена позволяют соответственно оценить затраты времени на непосредственное выполнение рабочего интервала и затраты времени на выполнение взаимодействий с рабочими интервалами других процессов.
В модели *HP* сохраняются нумерация вершин, дуг и их метки, заданные в модели поведения программы *H*.
Ограничим класс архитектур полносвязными однородными архитектурами. Данные ограничения позволяют задавать вычислительную сложность рабочего интервала временем его выполнения на процессоре и включать в это время затраты на взаимодействие с рабочими интервалами других процессов.

ПРИНЦИПЫ ПОСТРОЕНИЯ АЛГОРИТМА И ЕГО МОДИФИКАЦИИ

Алгоритм, описанный в [2], основан на пошаговом переходе от максимальной по числу процессоров ВС к минимальной. На первом шаге строится ВС с максимально возможным числом процессоров - каждый процесс программы назначается на свой процессор. Поскольку архитектура полносвязная, то в дальнейшем вместо процессоров будем рассматривать ассоциации процессов. Ассоциация процессов (SP_i) - упорядоченное множество рабочих интервалов процессов, назначенных на один и тот же процессор. На каждом последующем шаге происходит просмотр всех возможных пар ассоциаций, среди них выбирается наилучшая пара для обьединения, проверяется выполнение временных ограничений и выполняется коррекция числа ассоциаций. При пересечении времен выполнения рабочих интервалов ассоциации осуществляется их сдвиг таким образом, чтобы времена выполнения рабочих интервалов не пересекались. После объединения пары ассоциаций, количество ассоциаций будет уменьшено на единицу. На некотором шаге будет невозможно произвести объединение какой-либо пары ассоциаций без нарушения директивного срока или останется лишь одна ассоциация. На этом алгоритм заканчивает свою работу.
Схематично алгоритм можно представить следующим образом:
1. Начальная инициализация.
2. Выбор пары ассоциаций - кандидатов на объединение:
- *оценка значений пошаговых критериев оптимизации,*
- *проверка выполнения временных ограничений.*

3. Коррекция числа ассоциаций.
4. Нет ассоциаций, для которых возможно обьединение: завершение алгоритма.

Ниже рассмотрим и проведем анализ возможных вариантов операций алгоритма, пошаговых критериев оптимизации и способа их использования.
Операция начальной инициализации. Результатом выполнения операции является конфигурация исходной ВС: число ассоциаций равно числу процессов (в каждой ассоциации определены порядок выполнения и времена инициализации рабочих интервалов). Время выполнения программы и времена инициализации рабочих интервалов определяются следующим образом:

$T^w = \max_{i \in M} (T_i)$ - время выполнения программы *PR*;

$T_i = \max_{p_j \in SP_i} (t_j^{ini} + t_j), i \in M$ - время выполнения процессов *i*-ой ассоциации;

$t_j^{ini} = \max_k (t_k^{ini} + t_k)$ $k : \prec_{jk} \in \prec$ - время инициализации *j*-го рабочего интервала. Время инициализации рабочего интервала зависит от времени окончания работы всех его предшественников или $t_j^{ini} = 0$, если рабочий интервал не имеет предшественников и соответствует входной вершине графа *H*. Поскольку внутри одного процесса определен строгий порядок выполнения рабочих интервалов, то минимальное время инициализации определяется однозначно по данным формулам.

Операция выбора кандидатов на объединение. Результатом операции являются номера ассоциаций - кандидатов на обьединение. Для оценки кандидатов на обьединение необходимо ввести пошаговые (локальные) критерии оценки качества решения, получаемого при обьединении двух ассоциаций. Для пары обьединяемых ассоциаций используются следующие пошаговые критерии оптимизации:

1. Количество связей между процессами ассоциаций SP_a и SP_b. Введем множество пар рабочих интервалов предшественник-наследник, для ассоциаций SP_a и SP_b:
Links(SP_a,SP_b) = { $(p_i, p_j) : \exists \prec_{i,j} \in \prec$, ($p_i \in SP_a$ && $p_j \in SP_b$) || ($p_i \in SP_b$ && $p_j \in SP_a$) }. Тогда количество связей - это мощность этого множества:
NL(SP_a,SP_b) = | Links(SP_a,SP_b)|;
2. Количество рабочих интервалов, объединяемых в одну ассоциацию:
NW(SP_a,SP_b) = | SP_a | + | SP_b |;
3. Значение времени выполнения программы для нового набора ассоциаций:
$ET=T^w$;
4. Суммарное увеличение значений времен инициализации рабочих интервалов в ассоциации SP_c, полученного в результате объединения ассоциаций SP_a и SP_b :
$ST = \sum_{p_{c,i} \in SP_c} [t_{ci}^{ini} - t_x^{ini}]$,

где p_x - рабочий интервал из множества SP_a U SP_b, соответствующий $p_{c,i}$ из множества SP_c.

По первым двум критериям ищутся кандидаты с максимальным значением данного критерия, а по двум последним - с минимальным. Критерии можно применять в различных сочетаниях. Выбор из множества возможных кандидатов на объединение одной пары заключается в последовательном сужении множества возможных пар. На каждом шаге применяется один из критериев для оценки возможных кандидатов и множество возможных пар сужается до множества пар, на котором достигается минимальное/максимальное значение критерия. Если после последовательного применения всех критериев, множество возможных пар не удалось сузить до одной, то пара для объединения выбирается из них случайным образом. Следует также заметить, что значения первых двух критериев можно оценить без выполнения операции слияния, а последние два требуют ее выполнения. Соответственно, модификации алгоритма с использованием лишь первых двух критериев и проверкой временных ограничений лишь для выбранной пары, будут иметь значительно меньшую сложность, чем модификации алгоритма с двумя последними критериями или одним из них. В дальнейшем первые два критерия будем называть статическими, последнии два - динамическими.

Операция слияния. Результатом выполнения операции слияния является новый набор ассоциаций (на 1 меньший прежнего) и оценка времени его выполнения. Данная операция применяется при оценке значений динамических критериев и проверке временных ограничений. Операция слияния для выбранных двух ассоциаций $SP_a = \{p_{a1}, p_{a2}, \ldots p_{ak}\}$ и $SP_b = \{p_{b1}, p_{b2}, \ldots p_{bm}\}$ строит множество $SP_c = \{p_{c1}, p_{c2}, \ldots p_{c,k+m}\}$ для новой ассоциации SP_c. При этом требуется скорректировать времена инициализации всех рабочих интервалов про-

граммы так, чтобы не было пересечения времен выполнения рабочих интервалов в каждой ассоциации. Возможны три варианта построения множества SP_c :
1) множество SP_a переносится в SP_c без перестановок и далее в цикле происходит вставка по одному рабочему интервалу из множества SP_b, так чтобы SP_c оказалось упорядочено по времени инициализаций, и на каждой итерации происходит коррекция времен инициализаций (включая и SP_b), для того чтобы не было пересечений времен выполнений рабочих интервалов во всех ассоциациях;
2) в цикле происходит добавление по одному рабочему интервалу из исходных множеств, при этом из SP_a U SP_b берется еще не перенесенный рабочий интервал с наименьшим временем инициализации, в случае равенства берется рабочий интервал с наименьшим временем выполнения и на каждой итерации происходит коррекция времен инициализаций (включая SP_a и SP_b);
3) SP_c получается в результате упорядочивания SP_a U SP_b по значению времен инициализаций, но в отличие от второго варианта коррекция времен инициализаций происходит только после полного построения SP_c .
Операция коррекции числа ассоциаций. Количество ассоциаций уменьшается на единицу, рабочие интервалы двух ассоциаций объединяются в одну (используется операция слияния).
Различные модификации алгоритма могут быть получены:

- использованием различных операций слияния;
- изменением набора и/или порядка применения используемых критериев пошаговой оптимизации.

РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ АЛГОРИТМА МЕТОДАМИ ВЫЧИСЛИТЕЛЬНОГО ЭКСПЕРИМЕНТА

Вычислительные эксперименты по синтезу полносвязных архитектур показали высокое качество решений, получаемых алгоритмом, и возможность уменьшения вычислительной сложности алгоритма при сохранении качества получаемых решений следующим способом. На заданном числе начальных шагов пара для обьединения выбирается случайным образом, что исключает просмотр всех возможных пар ассоциаций на первых шагах, где их количество велико.
На рисунках 1-2 приведен тестовый пример для небольшого графа. На рис.1 показано распределение процессов по процессорам после выполнения операции начальной инициализации: каждый процесс выполняется на своем процессоре. На рис.2 показан результат работы алгоритма при использовании лишь одного пошагового критерия оптимизации - *NW*. В этом случае алгоритм получил оптимальное решение: для выполнения программы за директивный срок достаточно двух процессоров, загрузка процессоров равна 100%.
Тип операции слияния оказывает влияние на качество получаемого решения. Однако, это влияние не столь значительно по сравнению с влиянием набора и порядка применения пошаговых критериев оптимизации. Случайный выбор пар для слияния на начальных шагах позволяет уменьшить вычислительную сложность, при сохранении качества получаемого решения. Для плохого набора и порядка критериев, случайный выбор пар на начальных шагах позволяет даже значительно улучшить решение.

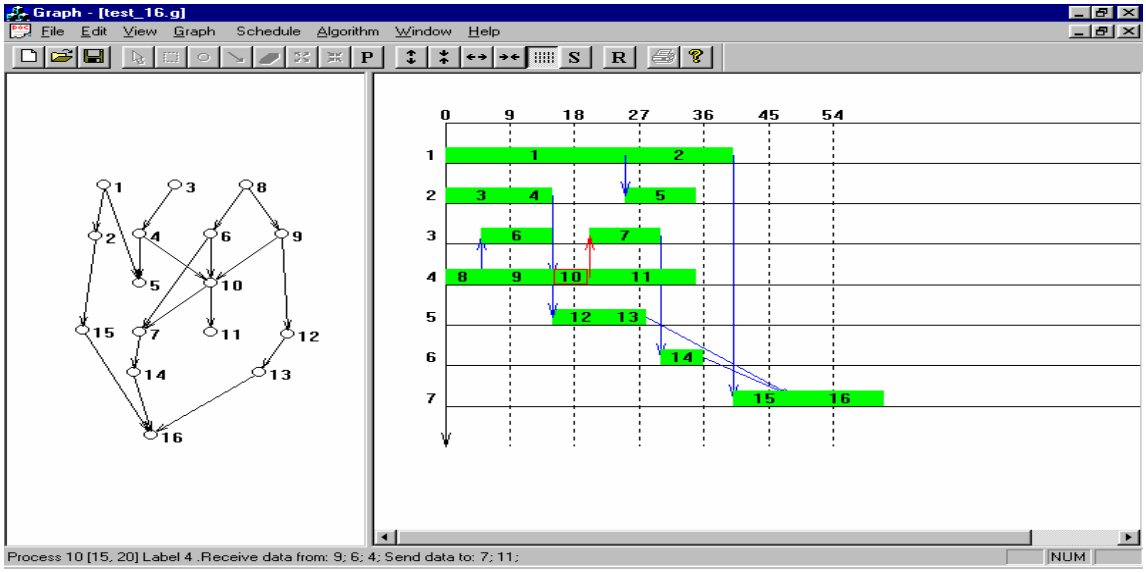

Рис.1. Каждый процесс выполняется на своем процессоре. Суммарное время последовательного выполнения - 160 тактов, критический путь - 61 такт, (вершины 1,2,15,16), директивный срок - 80 тактов.

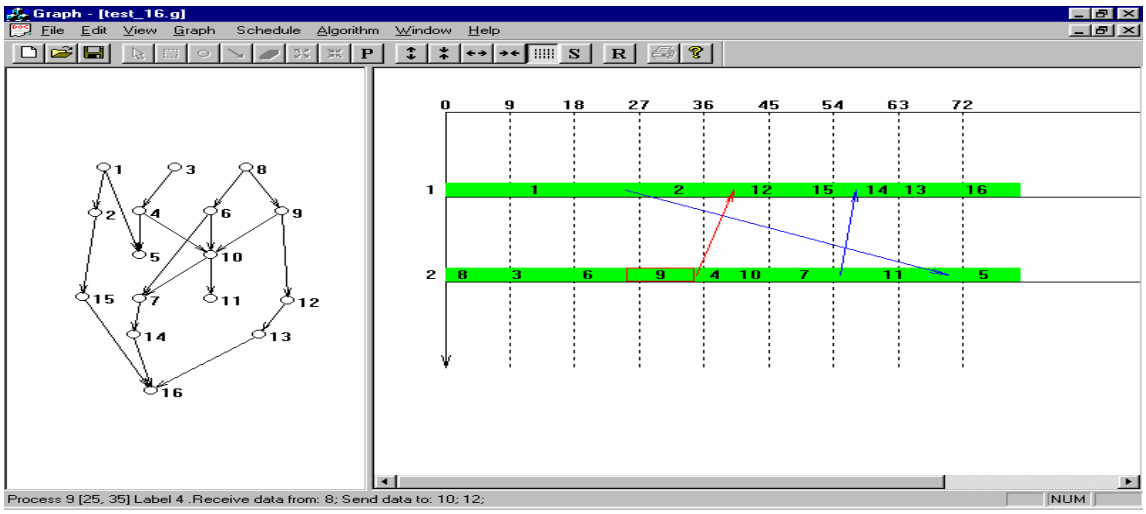

Рис.2. Критерии: *NW*, тип операции слияния: 3.

Недостатком данного подхода является сильная зависимость качества получаемых решений от набора и порядка использования пошаговых критериев оптимизации. Как показали предварительные исследования, устранить этот недостаток возможно одним из следующих способов: 1)использование комплексного пошагового критерия оптимизации; 2)осуществлять сужение множества возможных кандидатов на объединение не по строгому достижению минимального/максимального значения критерия, а по признаку принадлежности критерия некоторому интервалу в окрестности минимального/максимального значения критерия на текущем шаге; 3)разработкой методики настройки набора и порядка использования пошаговых критериев оптимизации на конкретное применение алгоритма. Анализу данных способов устранения зависимости качества получаемых решений от набора и порядка использования пошаговых критериев оптимизации будет посвящена отдельная статья.

ЛИТЕРАТУРА

1. Смелянский Р.Л. Об инварианте поведения программ// Вестн. МГУ, сер. 15, Вычислительная математика и Кибернетика, 1990., No. 4, С. 54-60.
2. Подгорный С.А., Смелянский Р.Л. К вопросу синтеза структуры вычислительных систем на основе поведения программ// Вопросы организации программного обеспечения и принятия решений. МГУ, 1993.