ОПРЕДЕЛЕНИЕ РЕСУРСА ПАРАЛЛЕЛИЗМА АЛГОРИТМА И ЕГО ЭФФЕКТИВНОГО ИСПОЛЬЗОВАНИЯ ДЛЯ КОНЕЧНОГО ЧИСЛА ПРОЦЕССОРОВ

Д.И. Свирихин

Цель работы состоит в создании аналитической методологии для эффективного распараллеливания процессов, выполняемых на вычислительных системах (ВС). В данном случае, предметом исследования является алгоритм в общем виде, а не какая-либо его реализация. Такой подход позволяет найти следующие характеристики алгоритма:

1. Максимально быстрая реализация алгоритма – определяет последовательность действий, позволяющую использовать весь доступный ресурс параллелизма алгоритма, необходимое количество для их выполнения процессоров и число тактов работы.
2. Максимально эффективная реализация алгоритма для n процессоров – определяет последовательность действий, наиболее эффективную для анализируемого алгоритма, для n процессоров. Получается путём оптимизации аналитических данных максимально быстрой реализации.

Для нахождения максимально быстрой реализации используется понятие Q-детерминанта, которое впервые было определено и формализовано в работе [1] В.Н. Алеевой. В основе данной концепции лежит преобразование последовательности действий алгоритма в Q-термы – выражения над множеством арифметических или логических переменных и множеством операций над этими переменными. В данной работе так же было описано, как с помощью алгоритма в форме Q-детерминанта отыскать максимально быструю реализацию алгоритма.

В работе [2] была доказана возможность преобразования алгоритма, описанного блок-схемой, в форму Q-детерминанта посредством применения конструкции под названием ActionList. Основная идея заключается в просмотре алгоритма с конечной точки его работы с целью поиска зацикливаемых и условных частей, поддающихся распараллеливанию.

Теория поиска сложностных характеристик и последовательсности действий максимально быстрой реализации алгоритма была рассмотрена в работе [3]. Для этого вводилось промежуточное представление алгоритма, именованное методом шаблонов, представляющее обобщённый метод записи Q-термов.

В настоящее время разработаны и реализованы программные средства для построения Q-детерминанта алгоритма, представленного в виде блок-схемы, а также для нахождения максимально быстрой реализации алгоритма.

Предполагается, что ВС удовлетворяет следующим условиям:

все процессоры системы идентичны и выполняют все базовые операции;

время выполнения операций на процессоре принимается за некоторое число тактов работы ВС;

если архитектура ВС с разделяемой памятью, то есть для каждого процессора выделена некоторая область памяти, указывается время пересылки, относительно времени такта работы ВС;

для архитектуры с единой памятью время выборки единицы информации из блоков памяти значительно меньше времени её обработки на процессоре, поэтому учитываться не будет.

На процессорах системы определено множество базовых и вспомогательных операций. В общем виде каждую из них можно представить следующим образом: Op(n,p,t,a), где n – арность (количество операндов), p – приоритет выполнения, t – время выполнения (количество тактов работы ВС), a – признак ассоциативности операции. Операция называется базовой, если она выполняется за один такт работы ВС. Вспомогательными называются операции, зависящие от базовых и выполняющиеся более одного такта работы.

Операнды On(lo) имеют значение только в контексте целого выражения. В таком случае каждый из них характеризуется уровнем вложенности операнда lo – величиной, определяющей номер такта работы, на котором операнд претерпел последнее изменение.

Выражение можно описать следующим образом: Exp(Op,On,le), где Op – множество операций, On – множество операндов, le – уровень вложенности выражения Exp, который определяется в процессе анализа выражения на предмет максимально быстрой реализации.

Данный анализ выполняется в несколько этапов:

1. Определение операций, выполнение которых возможно на i-ом такте работы ВС. Последовательно проверяются все операции выражения в порядке убывания приоритета.
2. Если

$$lo_{On_1} < i \quad \text{и} \quad lo_{On_2} < i$$

где

On_1 и On_2

некоторые операнды выражения, тогда операция Op над этими операндами выполнима на текущем такте работы.

3. После выполнения операции образуется новый операнд с уровнем вложенности

$$lo = \max(lo_{On_1}, lo_{On_2}) + t_{Op}$$

4. Выражение анализируется до тех пор, пока не останется единственный операнд. Уровень вложенности выражения le будет равен соответствующему значению этого операнда.

Алгоритм состоит из множества выражений, и уровня вложенности алгоритма, что можно представить, как

$$\alpha(Exp, l\alpha)$$

Уровень вложенности также определяется во время нахождения максимально быстрой реализации алгоритма.

Во время анализа необходимо учитывать следующие правила:

1. Если выражения

Exp_1 и Exp_2

являются независимыми, то их вычисление должно происходить одновременно.

2. Если

$$Exp_1$$

зависит от

$$Exp_2$$

то вычисление

$$Exp_1$$

необходимо начинать в тот момент, когда будет получен результат

$$Exp_2$$

То есть, ВС приступит к обработке

$$Exp_1$$

на

$$le_{Exp_1}$$

такте работы.

3. $$l\alpha$$

определяется максимальным достигнутым в ходе вычислений тактом работы ВС.

Последовательность выполнения операций для достижения максимально быстрой реализации можно обозначить с помощью множества деревьев действий [[f11.GIF]]. Пример такой последовательности для алгоритма скалярного произведения двух векторов [[f12.GIF]] и [[f13.GIF]] изображен на Рис. 1.

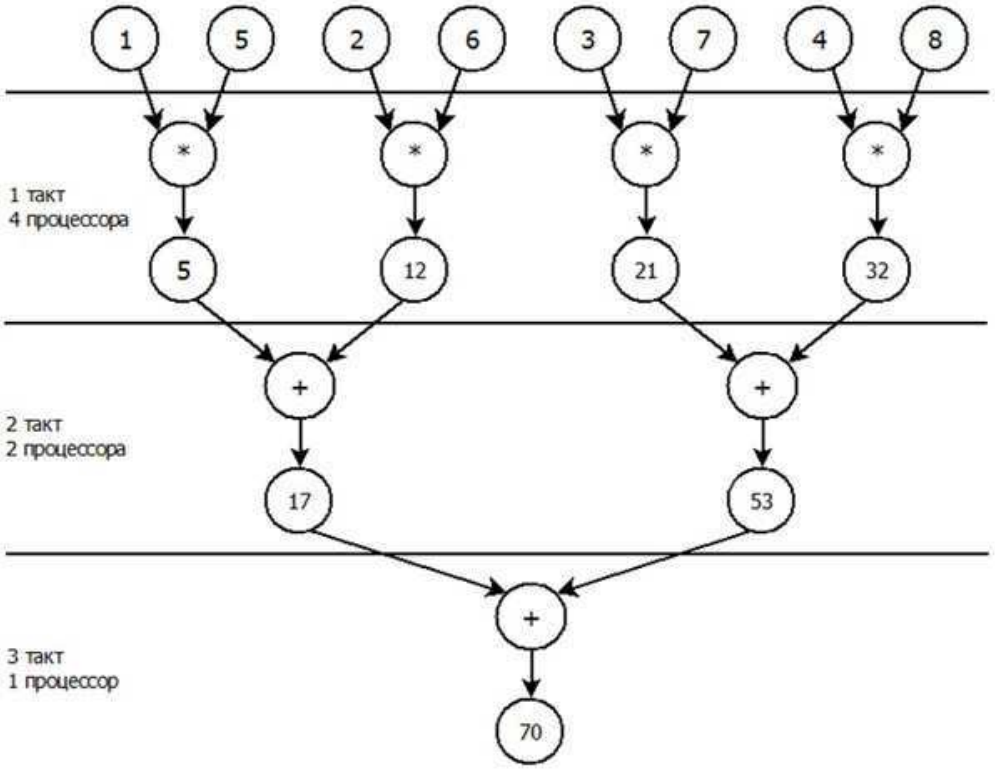

Рис. 1 Дерево действий максимально быстрой реализации алгоритма скалярного произведения двух векторов

Максимально эффективная реализация для n процессоров получается путем оптимизации $T(\alpha)$. Если, согласно $T(\alpha)$, коли $T(\alpha)$ чество операций на первом такте не превышает n, то максимально эффективная реализация будет полностью повторять максимально быструю.

Алгоритм оптимизации включает в себя следующие шаги:

1. Из i-ого такта работы $T(\alpha)$ выбираются m операций, где m – количество незанятых процессоров на j-ом такте работы оптимизированного алгоритма. Если i=1, то m=n.
2. Выбранные операции выполняются на j-ом такте работы оптимизированного алгоритма.
3. Повторяется шаг 1. Новые выбранные операции предназначены для выполнения на j+1 такте работы.
4. Шаги 1-3 повторяются до тех пор, пока на i-ом такте работы ВС не будут выбраны все операции.
5. Если условие шага 4 выполнено, то шаги 1-4 выполняются для i+1 такта работы ВС.
6. Оптимизированный алгоритм получится в тот момент, когда все операции $T(\alpha)$ будут выбраны.

На Рис. 2 приведен пример максимально эффективной реализации для трёх процессоров на рассмотренном ранее алгоритме скалярного произведения.

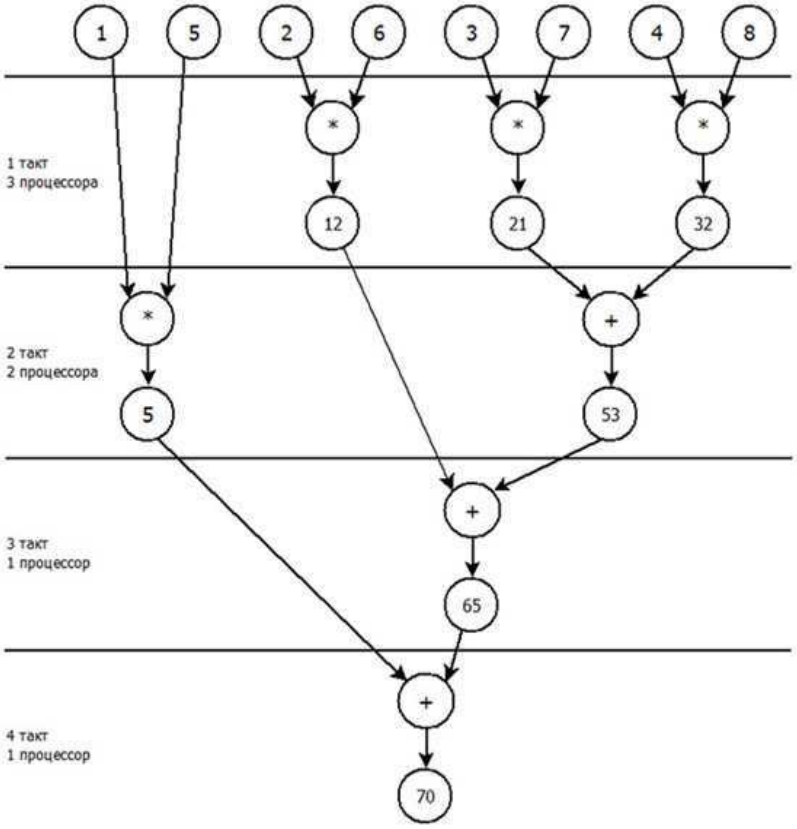

Рис. 2. Дерево действий максимально эффективной реализации алгоритма скалярного произведения двух векторов для трёх процессоров

Заключение

В настоящий момент в ходе работы были получены следующие результаты:

1. Разработан усовершенствованный метод анализа максимально быстрой реализации.
2. Разработан унифицированный способ использования ресурса параллелизма для конечного числа процессоров.

Предложенная методология является возможным решением для распараллеливания множества вычислительных задач. В дальнейшем планируется исследовать возможность отображения абстрактного дерева разбора на множество Q-термов. Это позволит производить оценку ресурса параллелизма алгоритм независимо от языка программирования, на котором он реализован.

ЛИТЕРАТУРА:

1. Алеева В.Н. Анализ параллельных численных алгоритмов: Препринт №590. Новосибирск, 1985. 23с. В надзаг.: ВЦ СО АН СССР.
2. Игнатьев С.В. Определение ресурса параллелизма алгоритмов на базе концепции Q-детерминанта: Вып. квалиф. работа ... магистра прикладной математики и информатики: 010500.68 / Южно-Уральский государственный университет. Челябинск, 2010. 75 л.
3. Свирихин Д.И. Программная система для определения ресурса параллелизма метода релаксации для решения СЛАУ: Вып. квалиф. работа ... бакалавра информационных технологий: 010400.62 / Южно-Уральский государственный университет. Челябинск, 2011. 37 л.