## Лабораторная работа №3 Распараллеливание рекурсивных схем

**Целью работы** является приобретение навыков по распараллеливанию рекурсивных арифметических выражений и оценке характеристик параллельности и эффективности решения.

Важнейшим классом арифметических выражений являются рекурсивные выражения. Рекурсия встречается в матричных операциях, является основой многих методов для вычисления значений функции и используется в ряде методов численного интегрирования.

При рекурсивных операциях значение каждого очередного члена последовательности или элемента структуры зависит от значений одного или нескольких предшествующих членов.

В последовательных ЭВМ рекурсия реализуется в виде циклических процессов. Рекурсия встречается не только при числовой обработке, но и при обработке сигналов, операциями над символами и т.п. Рекурсивный характер обработки накладывает ограничения на возможности распараллеливания, которое может быть достигнуто лишь реорганизацией последовательности обработки.

Остановимся на простейшем примере линейной рекурсии первого порядка для вычисления суммы  всех элементов вектора *X* = {*xi*}, где *i* =1, 2, …,n . В общем случае, в качестве операции может выступать любая ассоциативная операции. К числу таких операций относятся сложение, умножение, поиск максимума и минимума среди вещественных чисел, операции сложения и умножения матриц.

Для нахождения этой суммы можно воспользоваться следующим рекурсивным соотношением:

*Si* = *Si*–1 + *xi*, *i* =1, 2, …, *n*.

Последовательная реализация вычисления этого выражения иллюстрируется информационным графом на рис. 4, и поскольку каждый из операндов используется однократно, такой граф представляет собой бинарное дерево. Если выполнение каждой примитивной операции занимает время *t*, то суммарные затраты времени (при высоте графа *q*=n–1) реализации этого графа составят

*T*1(*n*) = *t* \* (n – 1) = *t* \* *q*.

Очевидно, что при наличии нескольких вычислителей алгоритм будет неэффективным, и задача состоит в том, чтобы минимизировать высоту дерева. Преобразование выполняется в соответствии с законами коммутативности, ассоциативности и дистрибутивности, в результате чего может быть получен трансформированный граф, показанный на рис. 5.

Высота такого дерева *q*'=[log2*n*], а длительность реализации составит *Tp*=*t*\*[log2*n*] в предположении, что операции суммирования реализуются за то же время *t*и затраты времени на обмен между отдельными вычислителями эквивалентны затратам времени на обмен между ячейкой памяти и вычислителем в последовательной структуре. Такой метод нахождения суммы элементов вектора получил название ***метода рекурсивного сдваивания*** или ***метода нахождения параллельных каскадных сумм***.

|  |  |
| --- | --- |
|  |  |
| Рис. 4. Прямое суммирование | Рис. 5. Попарное суммирование |

Максимальное ускорение *ξp*=*T*1**/***Tp* можно оценить как O(n/log2n). Оно достигается при числе вычислителей не менее n/2, причем полностью все вычислители загружены только на первом шаге вычислений, на последнем же шаге используется лишь один вычислитель.

Не используемые на каждом шаге вычислители можно отразить на графе в виде «пустых» операторов (заштрихованных кругов на рис. 5). Тогда эффективность построенной схемы алгоритма можно оценить как отношение числа действительных к общему числу операторов (действительных и пустых). Очевидно, что такое определение понятия эффективности сродни понятию коэффициента полезного действия. Для рассматриваемого алгоритма оно составляет 0.5 и не зависит от n.

Рассмотренный принцип рекурсивного сдваивания справедлив для произвольной ассоциативной бинарной операции на произвольном множестве.

Более общим случаем линейной рекурсии является вычисление значения полинома по схеме Горнера:

.

Отличительной особенностью схемы является чередование аддитивных (сложение, вычитание) и мультипликативных (умножение, деление, обратное деление) операций. Подобные выражения называют альтернированными.

Рекурсивный алгоритм для реализации этой схемы имеет вид:

 и *S*0 = *d*0.

Последовательная программа для нахождения *S* по этой схеме подобна приведенной выше для нахождения суммы, лишь команде сложения должна предшествовать команда умножения. Схема Горнера и рекурсивный алгоритм ее реализации весьма удобны для последовательных вычислений, однако их непосредственная реализация в параллельных вычислениях не приводит к ускорению, так как для выполнения очередной операции необходим результат, полученный на предыдущем шаге.

Известно, что альтернированные выражения всегда могут быть представлены путем разложения в виде функции нескольких арифметических выражений [3], например в виде:

*S* = *A* + *B*,

где каждое арифметическое выражение *A* и *B* содержит не более n арифметических операций, что доказывает возможность построения более быстрого параллельного алгоритма.

Поскольку основным ограничением для параллельной реализации рассматриваемой рекурсии является зависимость текущей операции от результата предыдущей, преобразуем схему вычислений так, чтобы вычисляемое значение *Si* на текущем шаге было связано с результатом, полученным не на предыдущем шаге, а на один шаг раньше, т.е. с *Si* – 2:



.

После подстановки второго выражения в первое получим:

.

Обозначив в этом выражении через  и , где верхний индекс означает номер последовательно применяемого описанного преобразования, получим:

.

Последнее выражение также является рекурсией, и к нему вновь можно применить это преобразование:



, где  и .

Последовательное применение указанного преобразования носит название циклической редукции, на каждом шаге которой получаем:

 и ,

где *k* – номер шага редукции.

В результате применения log2n шагов получим

*S* = *A* + *B*,

где , а 

Таким образом, нахождение значения полинома сводится к нахождению каскадной суммы произведений. Параллельный алгоритм для вычисления коэффициентов  аналогичен рассмотренному выше для нахождения сумм элементов вектора (операция сложения заменяется на операцию умножения).

Рассмотрим пример вычисления схемы Горнера для n=4.

, *S*0 = *d*0.

Используя формулы для циклической редукции, с учетом введенных выше обозначений, имеем:

,

.

Для данного примера дерево параллельного вычисления может быть представлено в следующем виде (см. рис. 6):



Рис.6. Дерево параллельного вычисления схемы Горнера для *S*4

Все арифметические операции сгруппированы по уровням; однотипные операции умножения или сложения, находящиеся в отношении безразличия, могут выполняться параллельно.

# 2. Подготовка к работе и порядок ее выполнения

Ознакомиться с теоретическим введением к данной работе.

1. Для вычисления значения полинома по схеме Горнера осуществить разложение схемы по методу циклической редукции. Построить вручную дерево параллельного вычисления. Определить характеристики сложности и параллельности. Вариант с четным номером выполняется для n = 8, с нечетным – для n = 7.

8. Проанализировать построенную параллельную схему для вычисления полинома с точки зрения оптимизации загрузки процессоров и, если необходимо, перестроить дерево вычислений. Определить характеристики параллельности полученного дерева вычислений.

9. Построить дерево параллельного вычисления схемы Горнера с помощью программы **expp.exe**, используя полученное в п.7 разложение и переименовав переменные в нем таким образом, чтобы они не повторялись.

# 3. Контрольные вопросы

1. Какие характеристики параллельности Вы знаете?
2. Чем характеризуется эффективность схемы при оптимальной загрузки процессоров?
3. К каким выражениям применяется схема сдваивания?
4. Какие выражения называют альтернированными?
5. Что лежит в основе распараллеливания рекурсивной схемы Горнера?
6. Какие основные свойства арифметических операций используются при преобразовании арифметических выражений в эквивалентные?

# 6. Список используемых источников

1. Системы параллельной обработки: Пер. с англ./ Под ред. Д. Ивенса. – М.: Мир, 1985. – 416 с.
2. Элементы параллельного программирования / Под ред. В. Е. Котова. – М.: Радио и связь, 1983. – 240 с.
3. Водяхо А.И., Горнец Н.Н., Пузанков Д.В. Высокопроизводительные системы обработки данных: Учеб. пособие для вузов. – М.: Высш. школа, 1997. – 304 с.
4. Математические модели и методы в параллельных процессах / Под ред. Воеводина В.В., 1986.
5. J.L. Baer and D.P. Bovet. *Compilation of arithmetic expressions for parallel computations*. In Proc. IFIP Congress, pp 340-346, Amsterdam, 1968.

*“Компиляция арифметических выражений для параллельных вычислений”*

1. R. P. Brent. *The parallel evaluation of arithmetic expressions in logarithmic time In Complexity of Sequential and Parallel Numerical Algorithms*, J. F. Traub, Ed, Academic Press, New York, pp. 83-102, 1973.
2. Richard P. Brent. *The Parallel Evaluation of General Arithmetic Expressions*. Journal of the ACM, v.21 n.2, pp. 201-206, 1974.
3. S. Winograd. *On the parallel evaluation of certain arithmetic expressions*. Journal of the ACM, v.22 n.4, pp. 477-492, 1975.