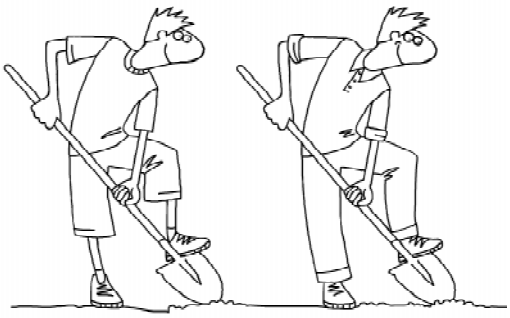
Демьянович Юрий Казимирович

ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ УРОК 1. ПАРАЛЛЕЛЬНАЯ ФОРМА

От редакции: в этом году курс «Заочной школы современного программирования» посвящен параллельным вычислениям и алгоритмам – мало освещенному в школьных учебниках, но очень перспективному направлению в развитии информатики и информационных технологий.

Термин «параллельные вычисления» в первую очередь вызывает представление о параллельных прямых, которые без дополнительных обозначений ничем не отличаются друг от друга. Естественно, что аналогичное представление о параллельных вычислениях не является правильным: бессмысленно проводить ничем не отличающиеся вычисления, ибо результаты окажутся, конечно, одинаковыми. Термин «параллельные вычисления» употребляется в другом смысле: речь идет об одновременно проводимых вычислительных процессах при решении одной задачи на компьютере. Отметим также, что слова «вычисления», «вычислительный процесс» используются не только для проведения арифметических действий, а во всех случаях взаимодействия человека с компьютером; сам компьютер часто называется вычислительной системой.



...которые без дополнительных обозначений ничем не отличаются друг от друга.

Для чего следует заниматься параллельными вычислениями при решении той или иной конкретной задачи? Конечно, в первую очередь, – для ускорения вычислений. Действительно, если задача распадается на несколько независимых подзадач, то для ускорения вычислений естественно решать каждую из них параллельно. Если эти подзадачи никак не связаны между собой, то есть представляют собой отдельные задачи, то нет необходимости проводить упомянутые вычисления на одном компьютере; такая ситуация не приводит к проблемам распараллеливания. Другое дело, если рассматриваемые подзадачи получают данные от предыдущей подзадачи, или результат их вычислений используется некоторой следующей подзадачей, или, наконец, они взаимодействуют друг с другом время от времени. В этом случае возникает проблема изолировать подзадачи друг от друга на период параллельного исполнения и обеспечить своевременные контакты между ними для обмена результатами их работы.

Из приведенных рассуждений следует, что организация параллельных вычислений является непростым делом. Возникает вопрос: стоит ли этим заниматься? Не лучше было бы для ускорения вычислений увеличить скорость работы процессора? Оказывается, что для увеличения скорости процессора осталось мало возможностей: увеличение скорости процессора требует уменьше-

ния его размеров, а с уменьшением размеров становятся все ощутимее законы квантовой физики, управляющие микромиром; эти законы носят вероятностный характер и не могут быть использованы на современном технологическом уровне. Именно это обстоятельство заставило пойти по пути создания параллельных вычислительных систем на множестве процессоров. Заметим, что появившиеся сравнительно недавно многоядерные процессоры фактически представляют собой многопроцессорные системы, реализованные в одном кристалле (в этой интерпретации ядра исполняют роль параллельных процессоров).

Идея распараллеливания вычислений появилась давно, поскольку представляется достаточно естественной для человеческого разума; первоначально она реализовывалась для локализации отдельных логических фрагментов в памяти вычислительной системы, с тем чтобы не было неконтролируемых взаимодействий этих фрагментов друг с другом. Примерами такого подхода служат операционная система Unix (1969 год; первыми ее авторами были Ken Tompson и Dennis Ritchie) и широко распространенные ее модификации (Red Hat, Fedora, OpenSuse, Mandriva и др.). Выполнение каждого из упомянутых фрагментов поручалось некоторому «процессу», в значительной степени отделенному от других «процессов». В те времена, конечно, не было параллельных вычислительных систем, компьютеры имели лишь один процессор, выполняющий арифметические и логические операции, поэтому процессы, объявляемые в Unix, выполнялись последовательно с соответствующими прерываниями – при переходе от исполнения одного процесса к другому. Заметим, что правильная организация прерываний – весьма сложная задача; фактически, это задача организации правильного времени исполнения различных участков программы с использованием памяти для хранения промежуточных результатов. Благодаря относительно большой скорости работы компьютера у пользователя появлялось приятное ощущение одновременности выполнения запущенных им процессов.

Накопленный здесь опыт широко использовался для создания других операционных систем и при разработке средств распараллеливания вычислений.

Параллельные вычисления, конечно, требуют определенного оборудования и программного обеспечения. Вместо одного процессора используется много процессоров (часто называемых параллельными вычислительными модулями), причем их количество варьируется от двух процессоров до десятков и сотен тысяч. Справиться с большим количеством процессоров, то есть загрузить их полезной работой – большая проблема. С одной стороны, не каждая задача распадается на параллельные подзадачи, а с другой стороны, в случае, когда число параллельных подзадач велико (например, тысяча параллельных подзадач), представляется нереальным составить для каждой из них компьютерную программу «вручную» и обеспечить их правильное взаимодействие; в таких случаях составляют одну программу, в которой предусматриваются условные переходы, определяемые номером (или именем) вычислительного модуля. При этом установленное на компьютере программное обеспечение обычно действует в достаточной степени самостоятельно. Оно распределяет работу между имеющимися вычислительными модулями, а если число последних меньше числа предлагаемых параллельных подзадач, то для некоторых таких подзадач запускает «виртуальные параллельные вычислительные модули», создавая впечатление, что эти параллельные подзадачи выполняются параллельно с остальными (на самом деле они, конечно, выполняются последовательно). Использование такого программного обеспечения позволяет программисту не заботиться о конкретных способах распределения подзадач по параллельным вычислительным модулям, ему даже не нужно знать, сколько параллельных модулей на данной вычислительной системе, каковы их характеристики (каково их быстродействие, каков объем памяти и т. п.). Благодаря этому упрощается работа программиста и поддерживается возможность перенесения написанной программы на другие вычислительные системы.

Программисту нужно решить основную проблему распараллеливания: указать группы подзадач (или операций), которые не зависят друг от друга и, следовательно, могут исполняться параллельно, а также указать последовательность выполнения этих групп. Решение этой проблемы приводит к параллельной форме: каждая группа операций называется *ярусом*, а число таких ярусов – *высотой* параллельной формы. Максимальное число операций в ярусах (число привлекаемых процессов в ярусах) называется *шириной* параллельной формы.

Один и тот же алгоритм может иметь много параллельных форм. Формы минимальной высоты называются *максимальными*.

Пример. Пусть требуется вычислить выражение

$$(a_1a_2 + a_3a_4)(a_5a_6 + a_7a_8),$$

соблюдая лишь тот порядок выполнения операций, который представлен в записи.

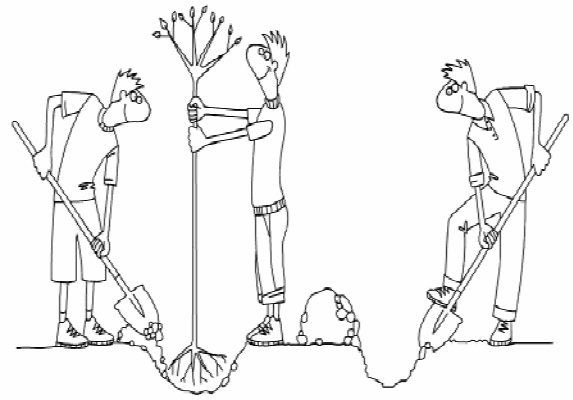
Здесь можно построить несколько параллельных форм, алгебраически эквивалентных по результатам.

1. Одна из них такова:

Данные: $a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8$.
Ярус 1. $a_1a_2,\ a_3a_4,\ a_5a_6,\ a_7a_8$.
Ярус 2. $a_1a_2 + a_3a_4,\ a_5a_6 + a_7a_8$.
Ярус 3. $(a_1a_2 + a_3a_4)(a_5a_6 + a_7a_8)$.

Высота этой параллельной формы равна 3, а *ширина* 4.

Особенность этой параллельной формы в том, что 4 процессора загружены только на первом ярусе, а на последнем ярусе загружен лишь один процессор.



...группы подзадач (или операций), которые... могут исполняться параллельно...

2. Вторая параллельная форма имеет вид:

Данные: $a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8$.
Ярус 1. $a_1a_2,\ a_3a_4$.
Ярус 2. $a_5a_6,\ a_7a_8$.
Ярус 3. $a_1a_2 + a_3a_4,\ a_5a_6 + a_7a_8$.
Ярус 4. $(a_1a_2 + a_3a_4)(a_5a_6 + a_7a_8)$.

Здесь для параллельных вычислений вполне достаточно двух параллельных процессоров.

3. Третья параллельная форма такова:

Данные: $a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8$.
Ярус 1. $a_1a_2,\ a_3a_4$.
Ярус 2. $a_1a_2 + a_3a_4,\ a_5a_6$.
Ярус 3. a_7a_8.
Ярус 4. $a_5a_6 + a_7a_8$.
Ярус 5. $(a_1a_2 + a_3a_4)(a_5a_6 + a_7a_8)$.

Ясно, что высота второй формы 4, третьей формы – 5, а ширина обеих форм одинакова и равна 2.

Для эффективного распараллеливания процесса стремятся

1) к увеличению загруженности системы процессоров;

2) к отысканию параллельной формы с заданными свойствами.

Существуют примеры, когда даже в простейших случаях (например, при реализации двойных циклов (см. [1: с. 385] и [2: с.126])) трудно ответить на вопрос, возможно ли распараллеливание, поэтому автоматизировать процесс распараллеливания (то есть исключить участие человека) в настоящее время не удается.

Как подходить к исследованию возможностей распараллеливания? Часто имеется много вариантов параллельных форм. Какую из них выбрать? Если возможно выполнять все операции каждого из ярусов за один такт работы параллельной системы, то, очевидно, число тактов будет минимальным для параллельной формы наименьшей высоты. Если набор рассматривемых параллельных форм исчерпывающий (то есть в этом наборе присутствуют все возможные параллельные формы), то меньшего числа тактов для выполнения рассматриваемой задачи получить не удастся. Поэтому возникает вопрос о минимальном числе тактов для решения данной задачи: если знать ответ на этот воп-

рос, то можно избежать напрасных исследований по поиску алгоритмов с меньшим числом тактов.

Для такого исследования удобно отказаться от предположения о том, что количество вычислительных модулей может быть меньше ширины рассматривемых параллельных форм. Такой подход, обогащенный предположением о том, что память вычислительной системы достаточно большая и быстрая, называется «концепцией неограниченного параллелизма».

В заключение предлагается ответить на следующие вопросы.

1. Что ознает термин «параллельные вычисления»?
2. Для чего заниматься параллельными вычислениями?
3. Как давно появилась идея параллельных вычислений?
4. Чем характеризуются средства для параллельных вычислений?
5. Что такое параллельная форма, каковы ее характеристики?
6. В чем состоит основная проблема, которую должен решать программист при распараллеливании вычислений?
7. Что такое «концепция неограниченного параллелизма»?

Попытайтесь также решить следующие задачи.

Задача 1

Пусть даны два ветора **a** и **b**. Точечным произведением **a** · **b** этих векторов часто называют вектор **c** = $(c_1, c_2, ..., c_k)$, компоненты которого получены по формуле

$$c_1 = a_1b_1, c_2 = a_2b_2, ..., c_k = a_kb_k.$$

Каковы возможности распараллеливания для отыскания результата точечного произведения **a** · **b**?

Существуют ли параллельные формы для его отыскания, а если существуют, то сколько их? Каковы их высоты? Какая форма имеет минимальную высоту?

Задача 2

Пусть требуется вычислить произведение восьми чисел $c = a_1a_2a_3a_4a_5a_6a_7a_8$.

Каковы возможности распараллеливания для отыскания результата c? Какие параллельные формы здесь можно предложить, сколько их? Какая форма имеет минимальную высоту?

Задача 3

Пусть требуется приближенно вычислить арифметический квадратный корень из положительного числа A. Один из способов такого вычисления – реализация итерационного процесса вида

$$x_{n+1} = \frac{1}{2}(x_n + \frac{A}{x_n}), n = 0, 1, 2, ..., N,$$

причем N определяется условием $|x_{N+1} - x_N| < \varepsilon$, а ε и x_0 – заданные положительные числа (о сходимости этого процесса см., например, в [3: с.7–9]).

Существуют ли параллельные формы для этого итерационного процесса, а если существует, то ответьте на прежние вопросы: сколько их, каковы их высоты, какая форма имеет минимальную высоту?

Литература

1. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. СПб, 2002. 608 с.

2. *Бурова И.Г., Демьянович Ю.К.* Алгоритмы параллельных вычислений и программирование. Курс лекций. СПб, 2007. 206 с.

3. *Иванов О.А.* Maxima в обучении математике. Урок 2. Компьютерные эксперименты с Maxima // Компьютерные инструменты в школе, 2010. № 2. С. 3–13.

Демьянович Юрий Казимирович, доктор физико-математических наук, профессор, заведующий кафедрой параллельных алгоритмов математико-механического факультета СПбГУ.