УДК 004.424.3::004.272.2::004.422.636

Новая технология параллельного программирования при решении некоторых задач с помощью процедур с повторным входом

Пекунов В.В., канд. техн. наук

Предлагается новый подход к программированию алгоритмов, сводимых к порождению серии схожих подзадач с планированием последовательности их решения в соответствии со стратегиями постановки в очередь, стек или дек. Предложена стратегия распараллеливания процесса решения задач на многоядерных системах с применением процедур с планированием повторного входа и цепи процедур.

Ключевые слова: процедура с планированием повторного входа, алгоритмы, параллельное программирование, очередь, стек, дек, цепи процедур, векторизация, конвейеризация, портфель задач, генератор.

The new approach to the parallel programming of the solving of some tasks using procedures with plan of reentering

Pekunov V.V., cand. tech. science

The new approach to programming of the algorithms reduced to generation of a series of similar subtasks with planning of sequence of their decision according to strategy of queue, stack or deque is offered. The approach proposes a new formalisms to include in high-level programming languages: procedures with plan of reentering and chains of them. Parallelization strategy of task decision on shared memory multicore systems is offered.

Keywords: procedure with plan of reentering, algorithms, queue, stack, deque, chains of procedures, parallel programming, vectoring, conveyorization, task pool, generator.

Одним из традиционных приемов программирования является сведение задачи к схожим подзадачам меньшей размерности. Существует серия основанных на таком подходе алгоритмов, предполагающих планирование последовательности решения подзадач на базе формализмов стека, дека, очереди. Назовем эти алгоритмы: а) поиска в ширину в дереве [1]; б) последовательной (по уровням) нумерации узлов дерева; в) поиска кратчайшего пути между электронными деталями на плате с применением волнового алгоритма Ли; г) нахождения в графе цепочки сетевых подграфов.

Классические решения с применением типовых шаблонных структур данных (реализованных, например, с помощью STL) требуют неявных ввода и контроля целого комплекса новых классов и явного ввода переменных, что усложняет код, повышая вероятность появления ошибок программирования, и негативно отражается на его плотности и скорости исполнения.

Планирование повторного входа в процедуру. Предлагается ввести в алгоритмические языки высокого уровня формализм процедур с планированием повторного входа, позволяющий более компактно и естественно представлять вышеназванные алгоритмы. Такие процедуры будут отличаться от обычных наличием плана исполнения, элементами которого (этапами) являются векторы значений параметров для очередного вызова процедуры. Спецификация синтаксиса процедур с повторным входом (void-функций C/C++) и связанных с ними конструкций доступна по ссылке http://www.pekunov.chat.ru/Progs.htm#Reenterable.

Динамический план. Любой явный (в том числе, рекурсивный) вызов процедуры с повторным входом создает *новый план*, который на начальном этапе содержит один элемент – вектор значений параметров, указанных при вызове процедуры. В ходе исполнения процедура может включать в начало или конец плана дополнительные этапы с указанием соответствующих значений параметров процедуры. При выходе из процедуры производится проверка плана: если план пуст, то осуществляется возврат в вызывающую программу, в противном случае из начала плана извлекается очередной этап, соответствующие ему значения параметров помещаются в параметры процедуры и производится повторный вход (переход в начало процедуры).

Статический план. Если процедура имеет *статический (постоянный) план*, то его исполнение может быть остановлено (с выходом из процедуры) и возобновлено при следующем входе в процедуру. Это позволяет разрабатывать *простые или рекурсивные генераторы* (разновидности сопрограмм), компактно реализующие, например, стек и очередь для сложных типов данных без явного ввода дополнительных структур.

Передача параметров. Редукция. Передача параметров по значению осуществляется обычным образом. Параметры, передаваемые по ссылке, по умолчанию «туннелируют» между вызовами: их последние значения с предыдущего этапа переходят на текущий этап (за исключением случая *отсечения*, где новое значение параметра для этапа указывается явно). Для такого параметра возможна *редукция*, когда к множеству его значений применяется некоторая бинарная коммутативная операция/функция, результат помещается в тот же параметр.

Применение процедур в параллельном программировании. Процедуры с повторным входом предполагают исполнение линейного плана этапов обработки. Предлагается представить план в виде последовательности непересекающихся групп этапов, разделенных специальными пометками (маркерами). Этапы группы могут выполняться либо последовательно, либо параллельно. *В параллельном режиме* группа рассматривается как динамически пополняемый «портфель задач» [3]. Можно ука-

зать количество процессоров, на котором исполняется процедура.

Перспективным представляется применение процедур с повторным входом для распараллеливания изначально рекурсивных алгоритмов, а также алгоритмов групповой обработки элементов рекурсивных структур данных (сетей, деревьев). Полная или частичная *замена рекурсии планированием* (итерацией) иногда дает столь же простое решение с генерацией меньшего количества потоков, как и при использовании Т-параллелизма [2].

Реализован ряд стандартных алгоритмов: поиска максимального элемента в дереве; сортировок слиянием и Шелла. Показана возможность достаточно эффективного *распараллеливания циклических алгоритмов с заранее неизвестным количеством не зависимых друг от друга итераций*.

Получаемый в результате применения предложенных конструкций код может иметь более явно выраженный параллелизм (векторный, конвейерный, портфель задач), легко выделяемый с помощью нескольких директив распараллеливания.

Цепи процедур с планированием повторного входа. Рассмотрим алгоритмы, имеющие последовательно зависимые (стадийные) или полностью независимые (параллельные) сегменты решения, каждый из которых предполагает генерацию и исполнение серии подзадач. Особый интерес представляют стадийные алгоритмы, в которых набор подзадач сегмента планирует множество подзадач следующего сегмента, предполагая иную последовательность их исполнения. Например, в волновом алгоритме поиска кратчайшего пути на первой стадии моделируется распространение волны и генерируется (в обратном порядке) план второй стадии, на которой найденный обратный путь неявно трансформируется в прямой. Сегментированные алгоритмы достаточно естественно могут быть реализованы *цепью процедур* с планированием повторного входа, в которой *каждая процедура реализует один из сегментов решения и может являться генератором плана для следующей процедуры в цепи*.

Векторный и конвейерный параллелизм в цепи процедур (автор выражает благодарность доценту Е.И. Ляпунову за идею применения процедур с повторным входом для конвейеризации расчета**).** Для реализации *векторного параллелизма* логичным решением является ввод механизма параллельного запуска процедур цепи. Аналогично реализуется *конвейерный параллелизм*: причем конвейерная передача данных реализуется путем вставки этапов в начало или конец плана следующей в цепи процедуры. Любая из процедур включается в работу, как только в ее плане появится по меньшей мере один этап работ. Процедура цепи может исполняться на нескольких процессорах («портфель задач»), а также запускать иные цепи. Таким образом реализуется *вложенный параллелизм*.

Реализованы векторный алгоритм умножения матриц и двустадийный конвеерный алгоритм поиска минимума и максимума среди элементов двоичного дерева (с внутренним распараллеливанием стадии).

Каналы. Для реализации более сложных алгоритмов параллельной обработки необходимы дополнительные, абстрагированные от архитектуры многопроцессорной системы средства обмена данными между процедурами, исполняемыми на различных процессорах (в разных потоках). Такими средствами могут быть *блокирующие типизированные каналы передачи данных*, аналогичные по функциональности введенным в языке MC# [4].

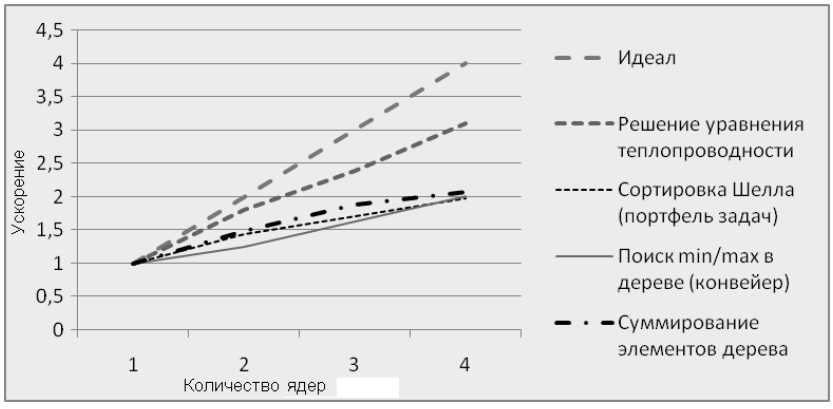
Реализация технологии. Разработан прототипный вариант препроцессора для языка C++ (http://www.pekunov.chat.ru/Progs.htm#Reenterable).

На рисунке показаны результаты замеров ускорения расчета для различных алгоритмов при работе с четырехъядерным компьютером (2×Opteron 270, 2 ГГц).

Планируется развитие подхода в целях дальнейшего повышения эффективности обработки рекурсивных структур данных, характерных, например, для нейронных сетей.

Заключение

Предложенный новый формализм процедур с планированием повторного входа дает более компактную и очевидную реализацию для ряда задач, традиционно решаемых с помощью вспомогательных типовых структур данных: очереди, стека и дека. В его рамках поддерживаются редукционные операции над параметрами процедур и возможна разработка аналогов простых и рекурсивных генераторов.



Ускорение, достигнутое при решении ряда задач на многоядерной системе

Предложенный механизм цепей процедур с повторным входом позволяет достаточно просто записывать некоторые алгоритмы, сводимые к решению серии подзадач с последовательной (стадийной) или параллельной (векторной) обработкой данных.

Разработанная новая компромиссная технология быстрого распараллеливания ряда алгоритмов менее громоздка по сравнению с традиционными средствами (OpenMP, DVM) и порождает меньшее количество потоков, хотя, возможно, и не столь эффективна, как программирование с применением семафоров и блокировок.

Список литературы

1. **Рассел С., Норвиг П.** Искусственный интеллект: современный подход. – М.: Вильямс, 2007.

2. **Воеводин В.В., Воеводин Вл.В.** Параллельные вычисления. – СПб.: БХВ-Петербург, 2002.

3. **Эндрюс Г.Р.** Основы многопоточного, параллельного и распределенного программирования. – М.: Вильямс, 2003.

4. **Петров А.В., Гузев В.Б.** MC# – универсальный язык параллельного программирования // Информационные технологии. – 2008. – № 4. – С.29–32.

Пекунов Владимир Викторович,
Ивановский государственный энергетический университет,
кандидат технических наук, доцент кафедры высокопроизводительных вычислительных систем,
e-mail: pekunov@mail.ru