Трещев И. А.
I. A.Treshchev

МЕТОДЫ ПОСТРОЕНИЯ СИСТЕМ АВТОМАТИЗИРОВАННОГО РАСПАРАЛЛЕЛИВАНИЯ ПРИЛОЖЕНИЙ ДЛЯ АРХИТЕКТУР С СИММЕТРИЧНО АДРЕСУЕМОЙ ПАМЯТЬЮ

METHODS FOR BUILDING SYSTEMS OF AUTOMATED PARALLELIZATION PROGRAMMING FOR ARCHITECTURES WITH SYMMETRICALLY-ADDRESSABLE MEMORY

Трещев Иван Андреевич – кандидат технических наук, доцент кафедры «Математическое обеспечение и применение ЭВМ» Комсомольского-на-Амуре государственного технического университета (Россия, Комсомольск-на-Амуре); 681013, г. Комсомольск-на-Амуре, ул. Ленина, д.27; 8-962-287-19-91. E-mail: kalkt@yandex.ru.

Ivan A. Treshchev - Ph.D in Engineering, Assistant Professor, Department of Computer Software and Computing, Komsomolsk-on-Amur State Technical University, 27, Lenina prospect, 681013 Komsomolsk-on-Amur, Khabarovsk region, Russian Federation, tel.: +7 (962) 287-19-91. E-mail: kalkt@yandex.ru.

Аннотация. Данная работа посвящена рассмотрению подходов для автоматизированного распараллеливания приложений, ориентированных на использование в системах с симметрично адресуемой памятью. Рассматриваются концепции параллелизма по данным и функционального параллелизма, модели с бесконечными и ограниченными ресурсами. Рассмотрена методология применения временных волновых систем для распараллеливания алгоритмов, которым присущ функциональный параллелизм.

Summary. The paper considers several approaches to automated parallelization of applications intended for use in systems with symmetrically addressed memory. Parallelization concepts are considered: data parallelization, functional parallelization; along with infinite/limited resource models . A methodology of application of time-wave systems to parallelize algorithms with inherent functional parallelism is considered.

Ключевые слова: автоматическое распараллеливание, функциональный параллелизм, параллелизм по данным.

Key words: automatic parallelization, functional parallelization, data parallelization.

УДК 681.3.06

Введение

В настоящее время широкую популярность приобрели микропроцессоры с симметрично адресуемой памятью, к ним относятся многопроцессорные вычислительные станции (два, четыре и более процессора, расположенных на одной материнской плате), многоядерные архитектуры (современные микропроцессоры Intel Core 2 Duo, Core 2 Quadro, Intel Xeon, Amd Opteron, Amd Athlon), системы, построенные по технологии NUMA. Для полноценного использования ресурсов современных вычислительных станций с SMP-архитектурой (Symmetric Multi Processing) актуальной является проблема создания приложений, поддерживающих параллелизм, присущий решаемой задаче и эффективно использующий ресурсы микропроцессора. Хотя второй закон Амдала [1, 67] утверждает:

$$\lim_{p\to\infty}\frac{T_1}{T_p}=\frac{1}{f},$$

Материал поступил 23.09.10

где f – доля операций выполняемых строго последовательно; T_1 – время исполнения на одном вычислительном узле; T_p – время исполнения на p вычислительных узлах, для каждой конкретной задачи достаточно сложно определить, какие операции могут исполняться последовательно, а какие можно исполнять одновременно. Созданию систем для автоматического выявления параллелизма посвящено достаточно много работ как отечественных, так и зарубежных авторов.

Параллелизм по данным

Пусть область входных данных алгоритма представляет собой объединение попарно непересекающихся подобластей, тогда на каждом из вычислительных узлов рабочей станции можно обрабатывать соответствующую подобласть независимо от остальных. Таким образом, происходит расслоение области входных данных на части, вычисления для которых можно производить независимо и одновременно.

Например, пусть нам необходимо построить на экране некоторое изображение, у нас имеется n процессоров (ядер), тогда мы можем разбить все изображение на n частей и на каждом из вычислительных узлов обрабатывать соответствующую часть одновременно, занося результаты обработки в некоторый буфер. По окончании обработки буфер будет отображен на экран.

Для систем с симметрично адресуемой памятью очень часто возникают задачи, которые связаны с использованием параллелизма по данным: сжатие информации, кодирование и декодирование видео- и аудио-, задачи машинной графики, шифрование, обработка текста, обработка данных в СУБД и т.д.

Применение стандарта OpenMP для создания приложений, поддерживающих параллелизм по данным, позволяет добиться ускорения выполнения приложений, но, в связи со сложностью освоения, требует дополнительных затрат времени на изучение данного стандарта, и не всегда его использование позволяет максимально эффективно задействовать все имеющиеся ресурсы.

Функциональный параллелизм

Рассмотрим вычисление значений функции, которая, в свою очередь, является суперпозицией некоторого числа элементарных функций, т.е. будем рассматривать функции вида:

$$g(x) = f_1(f_2(\ldots(f_n(x))\ldots),$$

где каждая из $f_i, i = \overline{1\ldots n}$ – элементарная функция.

Если множество входных данных довольно велико, то для вычисления каждой элементарной операции можно использовать свой процессорный элемент или отдельный поток.

Например, пусть нам необходимо вычислить значение функции $y(x) = \sin(\cos(tg(arctg(x)))$ для потока входных данных из 1 000 000 элементов, тогда необходимо задействовать четыре вычислительных узла (ядра), каждый из которых будет принимать на вход свой аргумент и на выходе – формировать значение, вычисленное после выполнения соответствующей элементарной функции. Следует отметить, что на синхронизацию действий, связанных с вычислением значений данной функции при использовании описанного алгоритма на системах с SMP-архитектурой, будет затрачено существенное для данной задачи время.

Для того чтобы максимально эффективно использовать ресурсы вычислительной станции при реализации функционального параллелизма, необходимо распределить по вычислительным узлам выполняемые элементарные функции так, чтобы не возникали ситуации, когда соответствующий узел не функционирует, ожидая на входе аргументов.

Отметим, что языкам функционального программирования Erlang, Prolog и другим, в связи с представлением программ в виде суперпозиции функций и использованием механиз-

ма backtrack, также присущ функциональный параллелизм, для которых методология построения многопоточных приложений описана в [5, 228; 6, 152;7].

Использование математических моделей последовательных процессов исполняющихся параллельно при автоматизированном построении приложений для SMP–архитектур

За последние 30 лет зарубежными исследователями Я. Фостером, Ч. Хоаром, Р. Милнером, К. Петри, Г. Винскелем, М. Нильсеном, Э. Гобо, М. Беднарчиком были предложены и изучены многие формальные модели, позволяющие описывать функционирование последовательных процессов, исполняющихся параллельно.

Также стоит отметить исследования в области построения формальных моделей распределенных вычислений отечественных ученых: В. Воеводина и Вл. Воеводина О. Омарова, В. Котова, И. Вирбицкайте, А. Барского, В. Бочарова, В. Корнеева, Н. Миренкова и др.

Наиболее общими математическими моделями параллельных вычислений являются: граф алгоритма, информационный граф алгоритма, граф зависимости и граф процесса [8, 21].

Многие классические модели: сети Петри, размеченные системы переходов, асинхронные системы переходов, автоматы высокой размерности [9, 100], графы зависимости, графы процесса, графы алгоритма и информационные графы алгоритма – позволяют исследовать проблемы, связанные с достижимостью некоторого состояния вычислительного процесса, его нетупиковостью, решать классические проблемы синхронизации [3, 98] (сериализации, взаимной блокировки, бесконфликтности), выявлять взаимосвязь моделей между собой, анализировать состояния таких систем различными методами.

Лишь немногие модели позволяют анализировать время функционирования таких систем, как временные сети Петри, временные структуры событий, временные системы переходов. Из них наиболее часто используемыми на практике являются временные раскрашенные сети Петри. Стоит отметить сложность реализации систем автоматизированного распараллеливания на основе этой модели в связи со сложностью теоретического анализа и ввиду асинхронной природы самих сетей Петри.

Одной из моделей, позволяющих анализировать время функционирования, является модель временных волновых систем, описанная в [4, 20].

Пусть задано некоторое арифметическое выражение. Узлы частичного орграфа волновой системы соответствуют примитивным операциям, а направленные ребра соединяют узлы, если результат выполнения операции, соответствующей началу ребра, является аргументом операции, соответствующей концу этого ребра. Из каждой вершины будет выходить столько ребер, для скольких последующих операций будет использоваться результат выполнения данной операции, и входить столько ребер, сколько аргументов она имеет. Каждой переменной арифметического выражения будет соответствовать вершина без входящих в нее стрелок. Результаты вычислений формируются в вершинах без выходящих из них стрелок. Причем введение операции параллельной композиции и последовательной композиции позволит оценивать время, необходимое для вычислений заранее.

Модель временной волновой системы эффективно применяется для распараллеливания алгоритмов, содержащих функциональный параллелизм.

С каждой вершиной временной волновой системы ассоциируем поток, а с каждым ребром – синхронизированную бесконечную очередь сообщений, что позволяет преобразовать вычисление арифметического выражения, заданного временной волновой системой, непосредственно в программу на языке высокого уровня, с использованием библиотек потоков для выбранной операционной системы.

Заключение

В современных операционных системах реализованы механизмы эффективного планирования выполнения задач и потоков, квитирования процессорного времени [2, 489], причем при наличии незанятого ядра очередной порождаемый поток будет исполняться на нем.

В настоящее время разработано программное обеспечение, которое позволит автоматизировать создание кода приложений для вычислительных станций с SMP-архитектурой, функционирующих под управлением современных операционных систем Windows и Unix, поддерживающих многопоточность и эффективно использующих их ресурсы. Данное программное обеспечение позволит эффективно использовать ресурсы вычислительной системы при разработке приложений, содержащих функциональный параллелизм и параллелизм по данным.

Также представляется возможным использование описанных в данной статье методик для построения распределенных приложений для систем с MPP-архитектурой (Massive Parallel Processing).

ЛИТЕРАТУРА

1. Воеводин, В. В. Параллельные вычисления / В. В. Воеводин, Вл. В. Воеводин. – СПб.: БХВ – Петербург, 2004. – 608 с.
2. Таненбаум, Э. Операционные системы: разработка и реализация / Э. Таненбаум, А. Вудхалл. // Computer Science: перевод с англ. – СПб.: Питер, 2006. – 576 с.
3. Топорков, В. В. Модели распределенных вычислений / В. В. Топорков. – М.: Физматлит, 2004. – 320с.
4. Трещев, И. А. Математическая модель гибридной временной волновой системы / И. А. Трещев // Системы управления и информационные технологии. – 2007. – № 4 (30). – С. 19–21.
5. Трещёв, И. А. Параллельные алгоритмы визуализации фрактальных множеств на основе асинхронных волновых систем / И. А. Трещев. // Информационно вычислительные технологии и их приложения: сб. ст. IV российско-украинского науч.-технического и методического симп. – Пенза: ПГСХА, 2006 – С. 227–230.
6. Трещев, И. А. Построение многопоточных приложений для распараллеливания алгоритмов перебора / И. А. Трещев. // Информатика и системы управления. – 2008. – № 1 (15). – С. 151–159.
7. Трещёв, И. А. Свидетельство об официальной регистрации программы для ЭВМ № 2006613475. Перебор последовательностей как раскраска вершин графа при обходе в ширину с использованием многопотоковых приложений на компьютерах с SMP архитектурой. – 6.10.2006.
8. Хусаинов, А. А. Архитектура вычислительных систем: учеб. пособие / А. А. Хусаинов, Н. Н. Михайлова. – Комсомольск-на-Амуре: ГОУВПО «КнАГТУ», 2004. – 123 с.
9. Bednarczyk, M. A. Categories of asynchronous systems, PhD thesis in Computer Science / M. A. Bednarczyk // University of Sussex. – 1988. – 134 с.