ВЕСТНИК ТОМСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
2018 Управление, вычислительная техника и информатика № 45

ИНФОРМАТИКА И ПРОГРАММИРОВАНИЕ

УДК 004.43
DOI: 10.17223/19988605/45/7

С.Б. Арыков

РЕШЕНИЕ ПРИКЛАДНЫХ ЗАДАЧ В СИСТЕМЕ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ АСПЕКТ

Экспериментальная система параллельного программирования Аспект предназначена для исследования новых форм представления алгоритмов и методов задания управления в параллельных программах. В статье кратко рассмотрены архитектура системы Аспект, особенности языка программирования Аспект, результаты применения системы Аспект для решения задач разложения матрицы, статистического моделирования методом Монте-Карло и моделирования методом «частицы-в-ячейках».
Ключевые слова: система параллельного программирования Аспект; фрагментированное программирование; представление алгоритмов с высокой степенью непроцедурности.

Современные суперкомпьютеры имеют сложную архитектуру. Они состоят из огромного количества вычислительных узлов, на каждом из которых есть несколько процессоров, каждый процессор содержит несколько ядер. Активно внедряются гибридные системы, где в дополнение к основным процессорам используются графические ускорители, специализированные вычислители типа Cell B.E., Matrix2000 или Intel Xeon Phi, вычислители на базе ASIC и PLD. Стремительный прогресс в использовании параллелизма аппаратной частью привел к большому разрыву между предоставляемыми ресурсами и возможностями по их эффективному использованию с помощью имеющихся средств разработки.

Для преодоления этого разрыва в России и за рубежом ведутся активные исследования по созданию средств параллельного программирования высокого уровня. К наиболее известным российским проектам можно отнести систему OpenTS [1] (Институт программных систем им. А.К. Айламазяна РАН), системы DVM [2] и НОРМА [3] (Институт прикладной математики им. М.В. Келдыша), язык mpC [4] (Институт системного программирования им. В.П. Иванникова), систему SFP [5] (Институт систем информатики им. А.П. Ершова), язык Пифагор [6] (Сибирский федеральный университет), систему ДВОР [7] (Южный федеральный университет), комплекс для параллельного программирования Graphplus templet [8] (Самарский государственный аэрокосмический университет). Результаты этих исследований показывают, что создание универсальной системы параллельного программирования высокого уровня трудно достижимо – слишком высока специфика отдельных предметных областей. Поэтому требуется дальнейшее изучение специализированных способов представления алгоритмов.

В статье представлены результаты исследований решения ряда прикладных задач в системе параллельного программирования Аспект. Аспект – это экспериментальная система, разрабатываемая в Институте вычислительной математики и математической геофизики СО РАН и предназначенная для исследования новых форм представления алгоритмов и методов задания управления в параллельной программе. В статье кратко рассмотрены архитектура системы Аспект, особенности языка программирования Аспект, результаты применения системы Аспект для решения задачи LU-разложения матриц, задачи оценки математических ожиданий аддитивных функционалов от траекторий диффузионных процессов и задачи о взаимодействии короткого лазерного импульса с плазмой методом «частицы-в-ячейках».

1. Система параллельного программирования Аспект

В основе системы параллельного программирования Аспект [9] лежат асинхронная модель вычислений с управлением на основе строгого частичного порядка [10] и фрагментированное программирование [11] как способ разработки параллельных программ. Архитектурно система Аспект состоит из транслятора с языка Аспект и исполнительной подсистемы.

Суть фрагментированного программирования заключается в представлении алгоритма решения задачи в виде множества *фрагментов данных* и множества *фрагментов кода*. Фрагмент кода получает на вход набор входных фрагментов данных, на основе которых вычисляет набор выходных фрагментов данных. Подстановка фрагментов данных в качестве параметров фрагмента кода называется *применением* фрагмента кода к фрагментам данных (один и тот же фрагмент кода может применяться к различным фрагментам данных). Совокупность фрагмента кода и его входных и выходных фрагментов данных называется *фрагментом вычислений*. На множестве фрагментов вычислений задается частичный порядок, т.е. множество ограничений на порядок выполнения фрагментов (управление).

Для реализации фрагментированного программирования на практике разработан язык управления вычислениями Аспект [12]. В нем фрагменты кода и фрагменты данных записываются на существующем процедурном языке программирования (в настоящее время поддерживается только C++), а фрагменты вычислений и частичный порядок их исполнения задаются средствами языка Аспект. Фрагменты вычислений, для которых зависимости явно не заданы, рассматриваются как независимые. Подробная информация о способах задания управления в языке Аспект приведена в [13].

Фрагментированная программа преобразуется транслятором языка Аспект в код на языке C++ и включается в одноименный класс, который предоставляет интерфейс для добавления фрагментов вычислений в очередь исполнения, а также осуществляет учет необходимых параметров (количество исполняемых фрагментов, количество готовых к исполнению фрагментов и др.). Для каждого фрагмента данных создается отдельный тип, с использованием которого объявляются данные задачи (переменные класса). Для каждого фрагмента кода генерируется метод класса, а также две переменных: признак завершения и переменная синхронизации.

Исполнительная подсистема Аспект состоит из набора функциональных модулей: менеджер потоков (управляет распределением работы между процессорами (ядрами); обращается за очередной порцией работы к планировщику), менеджер памяти (управляет распределением памяти в системе, осуществляет ее выделение / освобождение для очередей и фрагментированных программ), планировщик (управляет набором приоритизированных очередей, реализованных в виде мониторов; имеет сложность планирования O(1)).

2. Разложение матриц

На примере задачи разложение квадратной матрицы A в произведение матриц L и U (LU-разложения) рассмотрим применение фрагментированного программирования для решения численных задач.

Постановка задачи. Пусть имеется матрица $A = (a_{ij})$, $i = 1 \ldots N$, $j = 1 \ldots N$. Ее разложение в виде $A = LU$, где $L = (l_{ij})$ – нижняя треугольная матрица, а $U = (u_{ij})$ – верхняя треугольная матрица с единицами на главной диагонали, может быть вычислено по формулам [14]:

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj} \quad (i \geq j), \tag{1}$$

$$u_{ij} = \frac{1}{l_{ii}} \left(a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \right), \; (i < j). \tag{2}$$

Фрагментация алгоритма. Матрица A собирается из фрагментов данных (подматриц) A[i][j]. Выделяются четыре фрагмента кода F1, F2, F3 и F4. Первый фрагмент кода обрабатывает фрагменты данных, расположенные на главной диагонали матрицы A, по формулам (1) и (2); второй – фрагменты данных справа от диагональных элементов по формуле (1); третий – фрагменты данных под диагональными элементами по формуле (2); четвертый пересчитывает внутреннюю матрицу по формуле (3):

$$x_{ij} = a_{ij} - \sum_k l_{ik} u_{kj} \cdot \tag{3}$$

Вычисления организуются по шагам. Первый шаг показан на рис. 1. Квадраты отображают фрагменты данных. Внутри каждого квадрата указано, какой экземпляр фрагмента вычислений его обрабатывает. Стрелками отображены зависимости между экземплярами фрагментов вычислений, а цифра в центре показывает порядок исполнения экземпляров.

Первым запускается на исполнение экземпляр фрагмента вычислений G1[0] (применение фрагмента кода F1 к фрагменту данных A[0][0]), после чего параллельно могут исполняться экземпляры G2[0][1], G2[0][2], G3[1][0] и G3[2][0]. Любой из экземпляров с порядком исполнения 3 может начать исполнение, как только завершатся оба экземпляра, от которых он зависит. Например, G4[0][1][2] может начать исполнение после завершения G3[1][0] и G2[0][2].

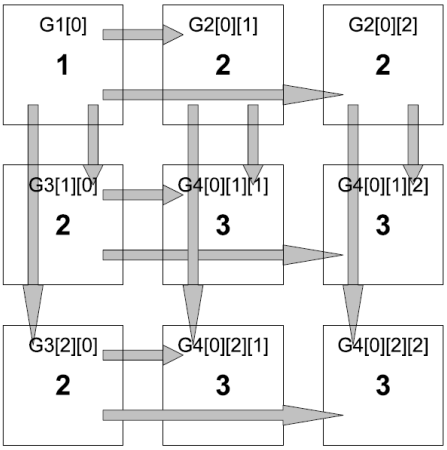

Рис. 1. Первый шаг вычисления LU-разложения матрицы *A*

Более подробно решение задачи LU-разложения матрицы разбирается в [11]. Текст программы LU-разложения матрицы на языке программирования Аспект можно найти в [13].

Результаты эксперимента. В качестве примера рассмотрим решение задачи о LU-разложении вещественной матрицы размера $5\,040 \times 5\,040$ при размере фрагмента данных 90×90 на системе HP ProLiant DL580 G5 (4 процессора Intel Xeon X7350, 16 ядер, 256 Гбайт RAM). Результаты представлены в табл. 1.

Таблица 1

Результаты измерений производительности программы решения задачи о LU-разложении

Характеристика	Количество ядер				
	1	2	4	8	16
Время счета, с	31,01	15,6	7,93	4,13	2,35
Ускорение	1	1,99	3,91	7,51	13,2

Аспект-программа демонстрирует практически линейное ускорение, что объясняется фрагментированным представлением алгоритма, за счет которого достигается более эффективное использование кэш-памяти. По этой же причине фрагментированный код лучше масштабируется.

3. Метод Монте-Карло

Статистическое моделирование является одним из численных методов решения математических задач, при котором искомые величины представляются вероятностными характеристиками какого-либо явления и приближенно определяются путем наблюдения за поведением модели этого явления [15]. Задачи статистического моделирования, как правило, требуют больших вычислительных ресурсов, поскольку в среднем выборочное значение может долго вычисляться и / или из соображений точности необходимо моделировать большое число выборочных реализаций.

Рассмотрим задачу оценки математических ожиданий аддитивных функционалов от траекторий диффузионных процессов, представляющих собой интегралы по некоторым малым областям расширенного фазового пространства от концентрации траекторий[1] [16]. На ее примере можно проследить, как в системе Аспект реализуются алгоритмы, которые можно фрагментировать на большое число независимых фрагментов вычислений.

Постановка задачи. Пусть трехмерный диффузионный процесс y(·) задан системой стохастических дифференциальных уравнений в смысле Ито:

$$dy(t) = a(y(t))dt + b(y(t))dw(t),\ 0 \le t \le T,\ y(0) = y_0, \qquad (4)$$

где $a(y)$ – трехмерная векторная функция, $b(y)$ – матричная функция размерности 3 × 3, $w(t)$ – трехмерный винеровский процесс, y_0 – случайный вектор.

Моделирование траекторий системы (4) выполняется с помощью метода Эйлера:

$$y_{i+1} = y_i + \Delta t a(y_i) + \sqrt{\Delta t} b(y_i)\xi_i,\ i = 0, 1, \ldots, n,$$

где y_i – численное решение системы (4) в узле t_i равномерной сетки с постоянным шагом $\Delta t = T / n$, а $\{\xi_i\}$ — последовательность независимых между собой нормальных случайных векторов с независимыми в совокупности компонентами, имеющими стандартное нормальное распределение.

Пусть необходимо вычислить полную концентрацию диффузионных траекторий $\Phi(T, y_c)$ во временном интервале $[0, T]$ в некоторой удаленной от источника точке y_c. При переходе к приближенной оценке точные диффузионные траектории заменим приближенными траекториями Эйлера, а точку y_c – шаром Ω_c малого радиуса r_c с центром в точке y_c. Тогда будем иметь следующую статистическую оценку:

$$\Phi(T, y_c) \approx E\zeta_D = E\sum_{i=0}^{n} \omega(y_i)\Delta t,$$

где $\omega(y) = \chi\Omega_c(y)/V_c$, V_c – объем шара Ω_c.

Фрагментация алгоритма. Разобьем всё пространство моделируемых траекторий на m областей. Тогда входные фрагменты данных – это начальные значения генератора случайных чисел в каждой области, выходные фрагменты данных – значение концентрации в каждой области.

Фрагментов кода необходимо два: первый вычисляет концентрацию в интервале $[j \cdot m, (j + 1) \cdot m - 1]$ (здесь j – номер части пространства моделирования), а второй осуществляет сложение всех концентраций, полученных на каждой области пространства моделирования, и вычисление окончательного результата.

Фрагменты вычислений получаются применением фрагмента кода первого типа к каждому входному фрагменту данных. Так как моделируемые реализации не зависят друг от друга, потоковое управление между этими фрагментами вычислений отсутствует. Все фрагменты вычислений первого типа должны выполниться до начала исполнения фрагмента вычисления второго типа.

Результаты эксперимента. В качестве примера рассмотрим систему (4) на временном интервале [0, 10] с постоянным вектором сноса $a \equiv (0, 0{,}5, 0{,}5)^T$, единичной матрицей диффузии $b \equiv I$, шаром Ω_c радиуса $r_c = 0{,}1$ с центром в точке $y_c = (8, 0, 0)$ и нулевым начальным условием $y_0 \equiv 0$ [17]. Для метода Эйлера положим $\Delta t = 10^{-2}$.

Таблица 2

Результаты измерений производительности программы оценки математических ожиданий аддитивных функционалов от траекторий диффузных процессов методом Монте-Карло

Характеристика	Количество ядер				
	1	2	4	8	16
Время счета, с	23 805,50	12 051,00	5 939,00	2 990,00	1 510,68
Ускорение	1	1,97	4	7,96	15,75

[1] Текст последовательной программы был предоставлен М.А. Марченко, ИВМиМГ СО РАН.

Результаты измерений на системе HP ProLiant DL580 G5 для 64 000 000 смоделированных траекторий приведены в табл. 2. Видно, что ускорение практически линейное, что достигается за счет независимого расчета концентрации в каждом интервале, а также хорошей локальности данных, используемых в ходе расчета.

Эта задача, тривиальная с точки зрения распараллеливания, тем не менее показывает, что сама система программирования Аспект не вносит существенных задержек и не требует существенных накладных расходов на исполнение фрагментов.

4. Метод частицы-в-ячейках

Методы частиц, как и конечно-разностные методы, образуют особую группу вычислительных алгоритмов, объединенных способом дискретизации исходной математической задачи [18]. Их применение широко распространено для решения сложных физических задач (например, задач физики бесстолкновительной плазмы). Вычислительная сложность метода объясняется огромным количеством частиц, необходимых для получения результатов с хорошей точностью.

Рассмотрим решение задачи о взаимодействии короткого лазерного импульса с плазмой методом «частицы-в-ячейках»[2] [19]. Она позволяет оценить применение системы Аспект для решения достаточно сложной задачи численного моделирования, включающей в себя несколько различных алгоритмов.

Постановка задачи. В области, имеющей форму параллелепипеда, находится фольга, представляемая тонким слоем плазмы. Плазма состоит из электронов и ионов одного типа. Лазерный импульс (в виде пакета электромагнитных волн различной амплитуды и поляризации) входит через поверхность параллелепипеда, взаимодействует с фольгой, частично отражаясь и проходя через нее.

Описываемый физический процесс представляется уравнениями Власова для ионов и электронов:

$$\frac{\partial f_{i,e}}{\partial t} + \vec{\mathrm{v}}\frac{\partial f_{i,e}}{\partial \vec{\mathrm{r}}} + \vec{\mathrm{F}}_{i,e}\frac{\partial f_{i,e}}{\partial \vec{\mathrm{p}}} = 0\,,\ \vec{\mathrm{F}}_{i,e} = q_{i,e}\left(\vec{\mathrm{E}} + \frac{1}{c}\left[\vec{\mathrm{v}},\vec{\mathrm{B}}\right]\right),$$

которые дополняются уравнениями Максвелла для электрического и магнитного полей:

$$\mathrm{rot}\,\vec{\mathrm{B}} = \frac{4\pi}{c}\vec{\mathrm{j}} + \frac{1}{c}\frac{\partial \vec{\mathrm{E}}}{\partial t}\,,$$

$$\mathrm{rot}\,\vec{\mathrm{E}} = -\frac{1}{c}\frac{\partial \vec{\mathrm{B}}}{\partial t}\,,$$

$$\mathrm{div}\,\vec{\mathrm{E}} = 4\pi\rho\,,$$

$$\mathrm{div}\,\vec{\mathrm{B}} = 0\,.$$

В свою очередь, плотность заряда ρ и плотность тока $\vec{\mathrm{j}}$ определяются из функций распределения ионов и электронов:

$$\vec{\mathrm{j}} = \sum_{i,e} q_{i,e}\int f_{i,e}\vec{\mathrm{v}}d\vec{\mathrm{v}}\,,$$

$$\rho = \sum_{i,e} q_{i,e}\int f_{i,e}d\vec{\mathrm{v}}\,.$$

Здесь индексы *i* и *e* обозначают тип частиц (ионы и электроны); $q_{i,e}$ – заряд; $f_{i,e}$ – функция распределения частиц.

В начальный момент времени частицы располагаются в области $x_0 < x < x_0 + \delta$ и образуют тонкую фольгу ($\delta \approx \lambda$, где λ – длина волны лазерного импульса). Известны линейные размеры и амплитуда волн лазерного импульса. Задана плотность плазмы. Скорости частиц, компоненты напряженностей электрического и магнитного полей равны нулю.

Методы и алгоритмы решения, а также дискретизация задачи детально описаны в работе [19].

[2] Текст последовательной программы был предоставлен В.А. Вшивковым, ИВМиМГ СО РАН.

Фрагментация алгоритма. Значения компонент напряженности электрического и магнитного полей, а также плотности заряда и тока дискретизованы на одинаковых (возможно, сдвинутых во времени и пространстве) сетках, каждая из которых собирается из фрагментов данных размера $f_i \times f_l \times f_k$. Информация обо всех частицах хранится в массиве фрагментов данных по f_j частиц в каждом фрагменте.

Один шаг моделирования выполняется с помощью пяти А-программ, которые содержат по несколько фрагментов кода и вычисляют магнитное поле (две А-программы), ток, электрическое поле и передвижение частиц. Вычисление электромагнитных полей и тока по каждому измерению выполняется отдельными фрагментами кода. Отдельные фрагменты кода обрабатывают вхождение импульса в пространство моделирования и границы этого пространства. Исполнение А-программ происходит последовательно, весь параллелизм сосредоточен внутри.

Фрагменты вычислений получаются применением фрагментов кода к соответствующим фрагментам данных сетки (электромагнитные поля и ток) либо к фрагментам данных массива частиц.

Результаты эксперимента. В качестве примера рассмотрим решение задачи со следующими параметрами. Фольга расположена на расстоянии 16 длин волны лазерного импульса λ от левого края области. Амплитуда лазерного импульса равна 20, длина и ширина импульса – 30λ и 10λ соответственно. Временной шаг τ составляет 0,01, число частиц одного заряда в ячейке – 32, общее число частиц – 10 485 760. Область разделена на 40 × 40 × 1 ячеек, размер пространства – 5 × 5 × 0,125.

Результаты измерений на сервере лаборатории синтеза параллельных программ ИВМиМГ СО РАН для 5 шагов моделирования приведены в табл. 3.

Таблица 3

Результаты измерений производительности программы моделирования взаимодействия короткого лазерного импульса с плазмой методом частицы-в-ячейках

Характеристика	Количество ядер				
	1	2	4	8	16
Время счета, без НТ	21,92	11,03	6,14	3,48	н/д
Ускорение, без НТ	1	1,99	3,57	6,30	н/д
Время счета, с НТ	21,92	18,68	10,0	5,1	3,57

Особенность задачи состоит в том, что расчет электромагнитных полей распараллеливается хуже (из-за интенсивного доступа к памяти), чем расчет движения частиц. Поэтому чем больше частиц используется при моделировании, тем лучше показатель ускорения. В то же время большее количество частиц приводит к лучшей точности.

Заключение

В статье рассмотрено применение системы параллельного программирования Аспект для решения ряда практических задач численного моделирования. Показано, что если задача допускает хорошее распараллеливание, она может быть эффективно реализована в системе Аспект без существенных накладных расходов на организацию исполнения фрагментов вычислений.

Результаты измерения производительности фрагментированных программ подтвердили гипотезу исследования: по высокоуровневому фрагментированному представлению алгоритма для выбранной предметной области возможна автоматическая генерация достаточно эффективных параллельных программ.

ЛИТЕРАТУРА

1. Абрамов С.М., Кузнецов А.А., Роганов В.А. Кроссплатформенная версия Т-системы с открытой архитектурой // Вычислительные методы и программирование. 2007. Т. 8, № 1. С. 175–180.

2. Бахтин В.А., Клинов М.С., Крюков В.А., Поддерюгина Н.В., Притула М.Н., Сазанов Ю.Л. Расширение DVM-модели параллельного программирования для кластеров с гетерогенными узлами // Вестник Южно-Уральского государственного университета. Сер. Математическое моделирование и программирование. 2012. № 18, вып. 12. С. 82–92.

3. Андрианов А.Н., Бугеря А.Б., Ефимкин К.Н., Колударов П.И. Модульная архитектура компилятора языка Норма+ // Препринты ИПМ им. М. В. Келдыша. 2011. № 064. 16 с.
4. Lastovetsky A. Adaptive parallel computing on heterogeneous networks with mpC // Parallel Computing. 2002. V. 28, No. 10. P. 1369–1407.
5. Kasyanov V.N., Stasenko A.P. A functional programming system SFP: Sisal 3.1 language structures decomposition // LNCS. 2007. V. 4671. P. 62–73.
6. Легалов А.И. Функциональный язык для создания архитектурно-независимых параллельных программ // Вычислительные технологии. 2005. Т. 10, № 1. С. 71–89.
7. Штейнберг Б.Я., Абрамов А.А., Алымова Е.В., Баглий А.П., Гуда С.А., Дубров Д.В., Кравченко Е.Н., Морылев Р.И., Нис З.Я., Петренко В.В., Полуян С.В., Скиба И.С., Шаповалов В.Н., Штейнберг О.Б., Штейнберг Р.Б., Юрушкин М. Диалоговый высокоуровневый автоматический распараллеливатель (ДВОР) // Труды Всероссийской суперкомпьютерной конференции «Научный сервис в сети Интернет» (Новороссийск, 20–26 сентября 2010 г.). М. : Изд-во МГУ, 2010. С. 71–75.
8. Востокин С.В., Литвинов В.Г., Хайрутдинов А.Р. Применение комплекса параллельного программирования Graphplus templet в моделировании // Программные продукты и системы. 2012. № 3. С. 14–18.
9. Arykov S.B., Malyshkin V.E. Asynchronous Language and System of Numerical Algorithms Fragmented Programming // LNCS. 2009. Vol. 5698. P. 1–7.
10. Арыков С.Б. Асинхронная модель вычислений над общей памятью // Вестник УГАТУ. 2016. Т. 20. № 3. С. 114–121.
11. Арыков С.Б. Фрагментированное программирование численных задач // Сборник научных статей VII Международной научно-практической конференции «Многоядерные процессоры, параллельное программирование, ПЛИС, системы обработки сигналов» (Барнаул, 10–11 марта 2017 г.). Барнаул : Изд-во Алт. ун-та, 2017. С. 20–24.
12. Арыков С.Б. Асинхронное программирование численных задач // Параллельные вычислительные технологии (ПаВТ’2010) : труды международной научной конференции (Уфа, 29 марта – 2 апреля 2010 г.). Челябинск : Изд. центр ЮУрГУ, 2010. С. 28–39.
13. Arykov S. Defining Order of Execution in Aspect Programming Language // LNCS. 2017. V. 10421. P. 265–271.
14. Фаддеев Д.К., Фаддеева В.Н. Вычислительные методы линейной алгебры. 2-е изд. М. : Наука, 1963. 656 с.
15. Михайлов Г.А., Войтишек А.В. Численное статистическое моделирование. Методы Монте-Карло. М. : Академия, 2006. 368 с.
16. Марченко М.А., Михайлов Г.А. Весовые алгоритмы статистического моделирования диффузионных процессов // Журнал вычислительной математики и математической физики. 2003. Т. 43, № 4. С. 571–584.
17. Марченко М.А. Комплекс программ MONC для распределенных вычислений методом Монте-Карло // Сибирский журнал вычислительной математики. 2004. Т. 7, № 1. С. 43–55.
18. Григорьев Ю.Н., Вшивков В.А., Федорук М.П. Численное моделирование методами частиц-в-ячейках. Новосибирск : Изд-во СО РАН, 2004. 360 с.
19. Вшивков В.А., Вшивков К.В., Дудникова Г.И. Алгоритмы решения задачи взаимодействия лазерного импульса с плазмой // Вычислительные технологии. 2001. Т. 6, № 2. С. 47–63.

Поступила в редакцию 18 апреля 2018 г.

Arykov S.B. (2018) SOLUTION OF APPLIED PROBLEMS IN THE PARALLEL PROGRAMMING SYSTEM ASPECT. *Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie vychislitelnaja tehnika i informatika* [Tomsk State University Journal of Control and Computer Science]. 45. pp. 59–67

DOI: 10.17223/19988605/45/7

Modern supercomputers have a complex heterogeneous architecture, which can include in various combinations CPU, GPU, DSP, ASIC, PLD, etc. It is difficult to efficiently use all these hardware resources with the available software tools for developing parallel programs, therefore further development of automated high-level parallel programming systems is an actual task.

Programming system Aspect is an experimental system of parallel programming developed at the Institute of Computational Mathematics and Mathematical Geophysics of the Siberian Branch of the Russian Academy of Sciences and designed to study new forms of representation of algorithms and methods for specifying control in a parallel programs. It is based on an asynchronous model of computation controlled by strict partial order and fragmented programming as a way of developing parallel programs. System Aspect consists of a translator from the Aspect language and the executive subsystem.

The essence of the fragmented programming is to represent an algorithm and its implementing program as a set of data fragments and code fragments. In the course of execution, the fragmented structure of a program is kept.

Each code fragment is supplied with a set of input data fragments (formal parameters) used to compute output data fragments. The substitution of data fragments as parameters into a code fragment is referred to as applying a code fragment to data fragments (the same code fragment may be applied to different data fragments). The code fragment with its input and output data fragments constitutes

a computation fragment. On the set of computation fragments, a partial order (control scheme) is defined. The resulting program is created from such computation fragments.

Aspect language for computations control was developed to implement fragmented programming on practice. In that language, code fragments and data fragments are defined in existing procedural programming language (currently only C++ is supported), while special constructs are used to define computation fragments and dependencies between them. The construction of control scheme is based on two key features: strict partial order and explicit support of massive control schemes. Computation fragments that do not have explicitly specified dependencies are considered independent and can be executed simultaneously.

Fragmented algorithms for solving LU-matrix decomposition problem, estimating the mathematical expectations of additive functionals from the trajectories of diffusion processes problem, and the interaction of a short laser impulse with plasma by the "particle-in-cell" method were developed and implemented in the Aspect system.

The results of experiments on the HP ProLiant DL580 G5 shared memory system (4 Intel Xeon X7350 processors, 16 cores, 256 GB RAM) showed that due to the locality of the data fragments and the efficient use of cache memory, fragmented programs allow achieving acceleration, close to linear, and provide good scalability. Experiments also showed that Aspect system does not introduce significant delays and does not require significant overhead for the execution of fragmented program.

Fragmentation of the algorithm may require its significant modification, and, therefore, cannot be performed automatically. This is a price for the good properties of fragmented programs.

Keywords: parallel programming system Aspect; technology of fragmented programming; representation of algorithms with high degree of nonprocedurality.

ARYKOV Sergey Borisovich (Candidate of Physics and Mathematics, Associate Professor, Institute of Computational Mathematics and Mathematical Geophysics of Siberian Branch of Russian Academy of Science, Novosibirsk State Technical University, Novosibirsk, Russian Federation).
E-mail: arykov@sscc.ru

REFERENCES

1. Abramov, S.M., Kuznetsov, A.A. & Roganov, V.A. (2007) Cross-platform version of T-system with open architecture. *Vychislitelnye metody i programmirovanie – Numerical Methods and Programming*. 8(1). pp. 175–180. (In Russian).
2. Bakhtin, V.A., Klinov, M.S., Kryukov, V.A., Podderyugina, N.V., Pritula, M.N. & Sazanov, Yu.L. (2012) Extension of DVM parallel programming model for clusters with heterogeneous nodes. *Vestnik Yuzhno-Uralskogo gosudarstvennogo universiteta: seriya "Matematicheskoe modelirovanie i programmirovanie" – South Ural State University Bulletin. Mathematical Modelling, Programming & Computer Software*. 18(12). pp. 82–92. (In Russian).
3. Andrianov, A.N., Bugerya, A.B., Efimkin, K.N. & Koludarov, P.I. (2011) Modular architecture of Norma+ language compiler. *Preprinty IPM im. M.V. Keldysha – Keldysh Institute Preprints.* 64. (In Russian).
4. Lastovetsky, A. (2002) Adaptive parallel computing on heterogeneous networks with mpC. *Parallel Computing*. 28(10). pp. 1369–1407. DOI: 10.1016/S0167-8191(02)00159-X
5. Kasyanov, V.N. & Stasenko, A.P. (2007) A functional programming system SFP: Sisal 3.1 language structures decomposition. *LNCS*. 4671. pp. 62–73.
6. Legalov, A.I. (2005) Functional language for creating architecturally independent parallel programs. *Vychislitelnye tekhnologii – Computational Technologies*. 10(1). pp. 71–89. (In Russian).
7. Steinberg, B.Ya., Abramov, A.A., Alimova, E.V., Bagliy, A.P., Guda, S.A., Dubrov, D.V., Kravchenko, E.N., Morylev, R.I., Nis Z.Ya., Petrenko, V.V., Poluyan, S.V., Skiba, I.S., Shapovalov, V.N., Steinberg, O.B., Steinberg, R.B. & Yurushkin, M. (2010) [Dialog high-level automatic parallelizer (DHAP)]. *Nauchnyy servis v seti Internet* [Scientific Service on the Internet]. Proc. of the All-Russian Supercomputer Conference. Novorossiysk, September 20–26, 2010. Moscow: Moscow State University. pp. 71–75. (In Russian).
8. Vostokin, S.V., Litvinov, V.G. & Khayrutdinov, A.R. (2012) Application of the parallel programming complex Graphplus templet in modeling. *Programmnye produkty i sistemy – Software and Systems*. 3. pp. 14–18. (In Russian).
9. Arykov, S.B. & Malyshkin, V.E. (2009) Asynchronous Language and System of Numerical Algorithms Fragmented Programming. *LNCS*. 5698. pp. 1–7. DOI: 10.1007/978-3-642-03275-2_1
10. Arykov, S.B. (2016) Asynchronous model of computation for systems with shared memory. *Vestnik UGATU*. 20(3). pp. 114–121. (In Russian).
11. Arykov, S.B. (2017) [Fragmented programming of numerical problems]. *Mnogoyadernye protsessory, parallel'noe programmirovanie, PLIS, sistemy obrabotki signalov* [Multi-core processors, parallel programming, FPGA, signal processing systems]. Proc of the Seventh International Conference. Barnaul, March 10–11, 2017. Barnaul: Altai State University. pp. 20–24. (In Russian).
12. Arykov, S.B. (2010) [Asynchronous programming of numerical tasks]. *Parallel'nye vychislitel'nye tekhnologii (PaVT'2010)* [Parallel computing technologies (PaVT'2010)]. Proc. of the International Conference. Ufa, March 29 – April 2, 2010. Chelyabinsk: South Ural State University. pp. 28–39. (In Russian).

13. Arykov, S.B. (2017) Defining Order of Execution in Aspect Programming Language. *LNCS*. 10421. pp. 265–271. DOI: 10.1007/978-3-319-62932-2_25
14. Faddeev, D.K. & Faddeeva V.N. (1963) *Vychislitel'nye metody lineynoy algebry* [Computational Methods of Linear Algebra]. 2nd ed. Moscow: Nauka.
15. Mikhailov, G.A. & Voishtek, A.V. (2006) *Chislennoe statisticheskoe modelirovanie* [Numerical statistical modeling. Monte-Carlo methods]. Moscow: Akademiya.
16. Marchenko, M.A. & Mikhailov, G.A. (2003) Weighted algorithms for statistical modeling of diffusion processes. *Zhurnal vychislitel'noy matematiki i matematicheskoy fiziki.* 43(4). pp. 571–584. (In Russian).
17. Marchenko, M.A. (2004) Kompleks programm MONC dlya raspredelennykh vychisleniy metodom Monte-Karlo [A set of programs MONC for distributed computing by Monte Carlo method]. *Sibirskiy zhurnal vychislitel'noy matematiki – Siberian Journal of Computational Mathematics.* 7(1). pp. 43–55.
18. Grigoriev, Yu.N., Vshivkov, V.A. & Fedoruk, M.P. (2004). *Chislennoe modelirovanie metodami chastits-v-yacheykakh* [Numerical simulation by particle-in-cell methods]. Novosibirsk: SB RAS.
19. Vshivkov, V.A., Vshivkov, K.V. & Dudnikova, G.I. (2001) Algorithms for solving the problem of interaction of a laser impulse with a plasma. *Vychislitelnye tekhnologii – Computational Technologies*. 6(2). pp. 47–63. (In Russian).