Министерство образования и науки
Российской Федерации
Федеральное агентство по образованию
Ярославский государственный университет
им. П. Г. Демидова
Кафедра математического моделирования

Параллельное и функциональное программирование

Методические указания

Рекомендовано
Научно-методическим советом университета
для студентов, обучающихся по специальности
Прикладная математика и информатика

Ярославль 2009

УДК 002:372.8
ББК З 973.2 – 018$_{я}$ 73
П18

Рекомендовано
Редакционно-издательским советом университета
в качестве учебного издания. План 2009 года

Рецензент
кафедра компьютерных сетей ЯрГУ

Составители Д. С. Глызин, Д. С. Кащенко

Параллельное и функциональное программирование: метод. указания /сост. Д. С. Глызин, Д. С. Кащенко; Яросл. гос.
П18 ун-т им. П. Г. Демидова. — Ярославль: ЯрГУ, 2009. — 28 с.

В методических указаниях приведена справочная информация, примеры и задания по курсу ”Современные компьютерные технологии”.

Предназначены для магистрантов первого курса, обучающихся по специальности 010501 Прикладная математика и информатика (дисциплина ”Современные компьютерные технологии”, блок ДС), очной формы обучения.

УДК 002:372.8
ББК З 973.2 – 018$_{я}$ 73

Содержание

1 Введение

Данные методические указания содержат начальные сведения по параллельному программированию систем с общей памятью на примере технологии OpenMP, а также базовую информацию по функциональному языку F#.

В настоящее время многоядерные и многопроцессорные компьютеры получили повсеместное распространение, поэтому для ресурсоемких задач необходимым требованием является максимально возможное распределение вычислений между ядрами.

Технология OpenMP (Open Multi-Processing) реализована для языков C и Fortran (мы рассмотрим ее лишь на примере первого) и предоставляет для создания параллельных программ как полноценный механизм планирования задач для каждого ядра, так и набор директив, позволяющих распараллеливать уже готовые последовательные программы, и таким образом в некоторых случаях увеличивать производительность программы на многоядерном компьютере без существенной переработки ее структуры.

Язык F# является адаптацией функционального языка OCaml под платформу .NET Framework. Функциональные языки предлагают более высокий уровень абстракции в сравнении с императивными и зачастую наиболее пригодны для решения задач математического характера.

Параллельное и функциональное программирование требуют во многом схожих навыков, не вполне типичных для наиболее популярной модели последовательного императивного программирования.

Многие параллельные алгоритмы, задействующие вложенный параллелизм, берут свое начало в методике "разделяй и

властвуй" и таким образом основаны на анализе рекуррентных зависимостей. И одновременно рекуррентные функции — это важнейший инструмент функционального программирования.

В качестве иллюстрации рассмотрим элементарный пример: пусть требуется вычислить сумму ста чисел. Последовательная модель подсказывает, что нужно начать с первого элемента, прибавить к нему второй, затем третий, и далее в цикле до самого последнего.

Естественным решением для параллельной реализации является разбиение множества суммируемых элементов на несколько подмножеств, после чего каждое подмножество обрабатывается своим процессором (ядром) в рамках последовательной модели.

Для функциональных языков, в которых переменные являются, по сути, константами, и потому не могут без дополнительных усложнений программы использоваться для хранения частичных сумм, естественным подходом является рекурсия. К примеру, функция может вызывать себя для подсчета суммы элементов первой половины множества, затем — для подсчета суммы элементов второй пловины, и сумму этих двух величин возвращать в качестве своего результата. При этом, в силу полной независимости по данным между "ветвями" рекурсивых вызовов, они могут быть выполнены разными процессорами.

Ключевой момент в двух последних случаях — умение программиста правильно разбить задачу на независимые части. В известной легенде о юном Карле Гауссе, согласно которой он быстрее всех в классе нашел сумму арифметической прогрессии $1, 2, \ldots, 100$ описан, по сути, пример исключительно продуктивной параллелизации вычислительной за-