Подготовка специалистов в области высокопроизводительных вычислений на базе межуниверситетской инновационной учебно-исследовательской лаборатории InterUniLab [1]

А.С. Абрамова, Н.А. Шехунова, А.В. Бухановский

Аннотация

Рассматриваются особенности разработки учебно-методического комплекса «Высокопроизводительные вычисления» на основе модульного и компетентностного подходов, ориентированных на слушателей-магистров по специализации «Разработка программного обеспечения» Санкт-Петербургского Государственного университета информационных технологий, механики и оптики. Самостоятельная работа в рамках комплекса ориентирована на участие в учебно-исследовательских проектах, выполняемых в межуниверситетской учебно-исследовательской лаборатории InterUniLab.

Введение

Современный этап развития высокопроизводительных вычислительных технологий характеризуется: широким распространением многоядерных компьютерных архитектур, удешевлением и доступностью кластерных систем на основе стандартных комплектующих, развитием технологий распределенных вычислений, в том числе, Грид [1]. Это требует модификации и развития соответствующих учебных комплексов. Недостаточное внимание сейчас уделяется системному подходу к параллельному математическому и программному обеспечению, как совокупности математических моделей, методов их реализации, параллельных алгоритмов, технологий программирования, тестирования и верификации параллельных программ, хотя именно такой путь позволяет строить эффективные параллельные алгоритмы и проектировать надежные программные системы на их основе [2].

Так как Россия в 2003 году присоединилась к Болонскому процессу, новые УМК должны разрабатываться в соответствии с требованиями, предъявляемыми Европейским союзом. Это позволит включить российские учебные курсы в европейскую систему образования. Главной проблемой перехода к новой системе образования, вызывающей полемику, является переход от квалификационного подхода к компетентностному, а также модульная структура обучения.

Преподавание высокопроизводительных вычислений (High Performance Computing, HPC), как дисциплины из области компьютерных наук, требует серьезной материальной и информационной базы. HPC быстро развивается, что приводит к необходимости постоянного обновления учебных материалов, которые должны в общем случае содержать мультидисциплинарные сведения (архитектура ЭВМ, теория построения алгоритмов, технологии программирования, коммуникационные технологии и пр.). Требования к педагогическому процессу в области высокопроизводительных вычислений, такие как направленность на конкретный результат образования и гибкость, дают право описать его в виде функциональной системы. Описание педагогического процесса в виде функциональной системы дает возможность эффективно управлять этим сложным процессом [3].

1 Разработка курса

Курс «Конструирование и анализ параллельных алгоритмов» разбит на модули: проектирование параллельных алгоритмов, проектирование параллельных программ, прикладные параллельные алгоритмы. По окончании курса студент должен уметь: строить эффективные параллельные

[1] Работа выполняется при финансовой поддержке Министерства образования и науки РФ в рамках приоритетного национального проекта «Образование».

алгоритмы, применительно к конкретной вычислительной архитектуре, уметь оценивать и моделировать параллельную производительность алгоритма для определенной вычислительной архитектуры, выбирать алгоритмы для решения поставленной задачи, выбирать технологии параллельного программирования для решения поставленной задачи, реализовывать ПО с помощью технологий параллельного программирования, оценивать эффективность работы параллельной программы и формулировать рекомендации по ее модификации.

Курс «Технологии распределенный вычислений и систем» разбит на 5 модулей: основные виды распределенных вычислительных архитектур, концепция и модели Грид, проектирование приложений для распределенных вычислительных архитектур, разработка приложений в peerto-peer-системах, разработка приложений в современных Грид-системах. По окончании курса студенты должны уметь выбирать оптимальный способ организации распределенной вычислительной системы, классифицировать распределенные вычислительные архитектуры, оценивать и моделировать параллельную производительность распределенной вычислительной системы, выбирать вычислительную систему для решения поставленной задачи, выбирать программный инструментарий для решения поставленной задачи, пользоваться программным инструментарием распределенных вычислительных систем, разрабатывать эффективное программное обеспечение для распределенный вычислительных систем.

В методическое обеспечение курса входит виртуальная лаборатория, которая формируется на базе разрабатываемых лабораторных работ. Лабораторные работы посвящены построению и оптимизации параллельных алгоритмов: метод Монте-Карло вычисления интегралов, решение систем линейных алгебраических уравнений методом Монте-Карло, генетический алгоритм (глобальная оптимизация), численное интегрирование (квадратуры), поиск на графах, умножение матриц, метод конечных элементов, параллельное III-разложение.

2 Работа студентов

Самостоятельная работа студентов в рамках данных курсов ориентирована на участие в учебно-исследовательских проектах, выполняемых в межуниверситетской инновационной учебно-исследовательской лаборатории InterUniLab. InterUniLab создана совместной инициативой Санкт-Петербургских университетов Санкт-Петербургского Государственного политехнического университета, СПбГУ ИТМО, Санкт-Петербургского Государственного университета авиаприборостроения и др. и Фондом содействия развитию малых предприятий в научнотехнической сфере при поддержке глобальных IT-компаний, таких как Intel, Microsoft, Cadence. Одной из ее задач является подготовка квалифицированных кадров в области критических технологий (в том числе, технологии распределенных вычислений и систем, высокопроизводительные вычисления) путем вовлечения слушателей в практическую реализацию мотивационных (курсовых) проектов индивидуального или в составе рабочей группы. Мотивационный проект ориентирован на разработку математического обеспечения высокопроизводительных вычислений в определенной предметной области. Слушателям на выбор будут предложены задачи из области гидрометеорологии, экологии, биомедицины, физики плазмы, технической диагностики и управления подвижными техническими объектами, основанные на реальных массивах данных.

УМК «Высокопроизводительные вычисления» СПбГУ ИТМО в настоящий момент находится в состоянии разработки. Однако отдельные его элементы уже прошли апробацию в рамках летних и зимних школ Intel (2006, 2007 гг.), а также в плановом учебном процессе СПбГУ ИТМО. Ввод УМК в опытную эксплуатацию планируется в осеннем семестре 2008 г.

3 Транслятор

Транслятор с языка программирования Аспект имеет классическую архитектуру и состоит из лексического анализа, синтаксического анализа контекстного анализа, генератора внутреннего представления и генератора кода. Реализация первых четырех модулей стандартна и далее не рассматривается.

На вход транслятору подается один или несколько файлов, в каждом из которых может быть одна или несколько A-программ, а также специальный конфигурационный файл, в котором для каждого массового A-блока указан требуемый размер A-фрагмента по каждой размерности. На выходе транслятор генерирует фрагментированную программу на языке C++, состоящую из одного главного файла (main.cpp) и нескольких заголовочных файлов по одному на каждую найденную A-программу.

Каждая A-программа представляется в виде класса, наследуемого от виртуального класса AProgram. A-программы могут быть вложенными (операция A-блока может реализовываться другой A-программой), и наличие единого предка позволяет унифицировать управление деревом A-программ.

Локальные информационные переменные представляются закрытыми (private) переменными класса, а входные/выходные информационные переменные — закрытыми ссылками. Все управляющие переменные также являются закрытыми. Для доступа к информационным переменным автоматически генерируются inline-функции доступа, которые помимо возврата данных могут производить проверку корректности обращения (например, проверку на выход за границы индекса).

Для каждой триггер-функции генерируется открытая (public) функция типа bool, а для каждой операции и управляющего оператора — соответствующие открытые процедуры.

4 Исполнительная подсистема

Исполнительная подсистема состоит из трех слоев: слоя абстрагирования от ОС, слоя функциональных модулей и слоя A-фрагментов.

Слой абстрагирования от ОС включается в себя все подпрограммы, в которых используются API операционной системы (создание/уничтожение/синхронизация потоков, функции определения доступных ресурсов, функции для работы со временем и т.д.). Это позволяет переносить исполнительную подсистему из одной ОС в другую заменой лишь одного, выделенного слоя.

Менеджер процессоров управляет распределением работы между процессорами (ядрами). При старте программы создается по одному потоку на каждый процессор (ядро). Каждый поток обращается за очередной порцией работы в общую очередь, которая реализована в виде монитора Хоара. Программа завершается, когда очередь становится пустой.

Менеджер памяти управляет распределением памяти в системе. С точки зрения фрагментированной модели вычислений его основной задаче является представление всех данных во фрагментированном виде, т.е. если массовый A-блок разбивается на n A-фрагментов по k экземпляров в каждом, то данные, обрабатываемые этим A-блоком, также хранятся во фрагментированном виде (n фрагментов по k элементов в каждом). Это позволяется при необходимости легко переносить один или несколько A-фрагментов с одного узла на другой (например, для динамической балансировки загрузки).

Менеджер коммуникаций отвечает за прием/передачу сообщений между узлами ЭВМ с распределенной памятью, однако в настоящее время этот модуль не реализован.

5 Результаты экспериментов

К настоящему моменту разработка системы программирования Аспект завершена не полностью и эксперименты по решению практических задач не проводились. Поэтому рассмотрим результаты тестирования ядра самой системы Аспект на примере модельной задачи непроцедурного умножения матриц (текст программы на языке Аспект приведен в разделе 3.2).

Для тестирования использовалась следующая конфигурация: Athlon $64X23600 + (2 * 256L2)$, $1024DDR2RAM$, Windows Vista Ultimate. Результаты тестирования для матриц размером 400 на 400 элементов приведены на рис. 4 (отметка Б/А показывает скорость аналогичной программы, разработанной вручную).

Из диаграммы видно, что ускорение достигает значения 1.93 (при размере A-фрагмента 50 экземпляров). Это объясняется тем, что хотя каждое ядро имеет отдельный кэш второго уровня,

доступ к оперативной памяти производится через общий контроллер. Таким образом, эффективное использование кэш-памяти для многоядерных процессоров имеет первостепенное значение.

Скачок при размере A-фрагментов в 400 экземпляров происходит потому, что его размер совпадает с исходным размером матриц, т.е. вся матрица представляет собой один A-фрагмент. В этой ситуации для второго ядра просто нет работы.

Следует обратить внимание, что ось A-фрагментов не линейна. Например, при размере A-фрагментов в 100 экземпляров всего получается 80 A-фрагментов ($4 * 4 * 4 + 4 * 4$), а при размере в 50 экземпляров — уже 576 A-фрагментов. Таким образом, при существенном увеличении числа фрагментов скорость счета не возрастает (она даже несколько снижается под влиянием кэшпамяти), что свидетельствует об эффективности работы ядра системы.

Список литературы

[1] Defining the Grid – a snapshot of the current view // H. Stockinger, 2006 (www.gridclub.ru) 1

[2] Гергель В.П., Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем. – Н. Новгород, ИНГУ, 2001 1

[3] Анохин П.К. Принципиальные вопросы общей теории функциональных систем. – М, 1973 1