О ПОСТАНОВКЕ КУРСА "ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ"

Л.В. Городняя

Аннотация

Доклад посвящен содержанию курса, нацеленного на ознакомление с моделями параллелизма, встречающимися в языках и системах программирования. Рассматриваются парадигмы параллельного программирования от уровня базовых средств управления взаимодействующими процессами до уровня разработки программ высокопроизводительных вычислений. Работа поддержана грантом РФФИ 08-01-00899а

1 Введение

Рост интереса к параллельному программированию связан с переходом к массовому производству и применению многоядерных архитектур. Появление информационных сервисов выплеснуло проблематику параллельных процессов на рядового пользователя. [1] Производительность, надежность и безопасность становятся ведущими показателями качества программ. Повышается роль такого критерия как понятность принципов параллельной обработки информации, что в значительной мере требует изучения механизмов взаимодействия процессов. С взаимодействующими процессами студенты встречаются при работе на уровне операционных систем (ОС), а также при организации практики на базе суперкомпьютеров, при сетевой обработке данных, при компиляции учебных программ и т.п. Проблемы подготовки программ для всех столь разных работ обладают общностью, но есть и существенная специфика, изучение которой и является основной задачей предлагаемого специального курса.

Сложность перехода от навыков последовательного программирования к организации параллельных процессов в значительной мере обусловлена системой обучения, требующей все упорядочивать, выстраивать в однозначные, безальтернативные построения, без выражения зависимости принятых решений от выбора структур данных и последовательности действий по их обработке. Т. Хоар утверждает, что "Параллельная композиция действий внешне не сложнее последовательного сочетания строк в языке программирования" [7].

2 Структура курса

Изложение проблем параллельного программирования в разрабатываемом курсе начинается с пояснений к терминологическим системам, сложившимся в разных коллективах, направлениях исследований и подходах к параллельным вычислениям. [1,4,7,9] Реализация процесса - функция, определяющая ход процесса по начальному событию. Таким событием может быть в частности готовность данных. Функциональная модель представления процессов позволяет легко описывать взаимодействие процессов в виде функционалов, т.е. функций над процессами, представленными как функциональные переменные. При определении взаимодействий используется понятие "протокол". Протокол - это последовательность символов, обозначающих произошедшие события. Важное направление анализа протоколов - проверка соответствия объектов спецификации процесса. Разработка программ для организации взаимодействия процессов отличается от подготовки обычных последовательных программ на весьма глубоком уровне. [7,9] Функциональный подход к параллельному программированию ценен возможностью унификации понятий, различие которых мало существенно с точки зрения их реализации.

Для программистов, не имеющих опыта функционального программирования, курс предлагает обзор основных идей современных языков, поддерживающих подготовку надежных программ на базе символьной обработки данных и программ, использующих списки и множества в качестве универсальных структур данных. [2] Рассматриваются простейшие методы оптимизации и преобразования программ и процессов, а также средства спецификации программ на базе формальных языков и сетевых граф-схем. В частности изучается сетевое представление семантики параллелизма на основе модели сетей Петри, приспособленной для алгебраического подхода к описанию процессов. [4]

При изложении материала естественно выделяются уровни определений и понятий, соответствующие уровню языков и систем программирования. На базовом уровне рассматривается управление процессами, изучаются средства и методы представления временных отношений с помощью очередей и приоритетов, проблема синхронизации и остановки. Параллельное программирование на уровне ОС выглядят как нечто среднее между программированием на макроассемблере и языках высокого уровня (ЯВУ). [3]

Основной уровень – программирование на ЯВУ. Рассматриваются версии ряда стандартных языков императивного программирования, приспособленные к выражению взаимодействия последовательных процессов в предположении, что в каждый момент времени существует лишь один процесс. Многие традиционные языки программирования (ЯП) приспособлены к выражению параллелизма с помощью специальных расширений или библиотечных функций, обеспечивающих выделение участков с независимыми действиями, пригодными для распараллеливания компилятором. [1]

Автоматическое распараллеливание последовательных программ ограничивает ускорение вычислений. Более успешным может быть выражение языковыми средствами параллелизма на уровне постановки задачи. В

таком случае при оптимизирующей компиляции возможен аккуратный выбор эффективной схемы параллелизма, что можно проиллюстрировать на моделях параллелизма, реализованные в наиболее известных и интересных ЯП, давших исторически ценный опыт представления параллельных вычислений. Рассмотрены языки APL, Algol-68, Setl, Ada и др. [10]

Имеется и небольшой экскурс в отечественные проекты в области языков параллельного программирования (БАРС, Поляр, Норма, mpC и др.). [1,4,5] Более подробно рассматриваются новые языки и системы, имеющие перспективу стать практичными средствами параллельного программирования в наше время (Microsoft .Net F# и C#, Haskell, Python и Sisal). [6,8] Анализируются проблемы разработки, отладки, тестирования и верификации параллельных программ.

3 Базовые средства параллельного программирования

Необходимость пояснений к терминам параллельного программирования обусловлена исторически сложившейся весьма пёстрой речевой практикой, дающей пищу техническим разночтениям общеизвестных понятий (поток, процесс, действие, операция и т.д.). Поэтому в курсе анализируется содержательная эквивалентность и различие общих терминов с тем, чтобы определить место таких понятий как условие готовности и срабатывание, завершение и оттеснение, время и длительность, пропускная способность и производительность, события и исключения, одновременность, ускорение вычислений.

Понятия "параллелизм" и "параллельные алгоритмы" рассматриваются с точки зрения оценки производительности систем. Поэтому необходим обзор парадигм программирования, показывающий разнообразие подходов к программированию вообще и к параллельному программированию в частности. Особое внимание уделено неимперативным вычислениям, преимущественно функциональному программированию, показавшему свои преимущества в параллельном программировании. Рассматривается техника символьной обработки разных структур данных, включая списки и множества, используемые при организации вычислений (очереди, протоколы и др.). Символьные вычисления рассматриваются как важный этап разработки эффективных и надежных программ и удобных ЯП. [2]

Общие идеи парадигм программирования дополняются кратким обзором средств и методов представления программ и процессов в форме определения языков и изображения эквивалентных им сетевых граф-схем с целью демонстрации возможностей декомпозиции параллельных программ и реорганизации процессов их выполнения, включая распараллеливание. Сетевое представление семантики параллелизма изучается в терминах сетей Петри, приспособленных к достаточно тонкой детализации семантики параллелизма и ее представлению в форме алгебры процессов и иерархических сетей, удобных для задания уровней определений и конкретизации понятий. Рассматривается проблема синхронизации независимых сетей. Представление сетевых и распределенных систем рассматриваются как обоснование к выбору единого сетевого представления семантики программ, процессов, языков и систем программирования. [4]

Базовый уровень определения параллельных процессов сводится к представлению временных отношений и взаимодействия процессов, включая управление процессами в сетях. Рассматривается время как ресурс. В качестве примеров используются автоматы Т.Хоара, при программировании которых используются потоки, счетчики, барьеры. [7]

Объектно-ориентированный подход к программированию сделал популярным представление программ в виде компонент, взаимосвязанных с помощью событий и сообщений. Рассматривается техника декомпозиции программ на такие компоненты и дополнения программ обработчиками исключений. Используются в качестве примеров задачи на клетчатой доске (взаимодействие процессов). Управление памятью рассматривается как доминирующая линия в определении семантики программ. [6,7]

Изучаются подходы к определению условий готовности компонент к срабатыванию, пространств имен как инструмента синхронизации, бесконечные и пошаговые процессы. Рассматриваются проблемы организации асинхронных процессов, возникающие при разработке распределенных систем, ленивые и энергичные вычисления, синхросети и мониторы. (Иллюстративные примеры: задачи о философах, читателях-писателях и т.п.) Представление сетевых и распределенных систем рассматриваются как основа для разработки сложных информационных систем и ЯП.

4 Типичные средства параллельного программирования

Практика параллельного программирования преимущественно сосредоточена на задачах, сводимых к векторной обработке информации, решаемых техникой распараллеливания отчасти модернизируемых последовательных программ, не учитывающих особенности работы с многоуровневой памятью и специфику типов управления в многоядерных архитектурах. В этой связи изучаются модели параллелизма для матричных вычислений, идеальный параллелизм без ограничения числа процессоров над одноуровневой общей памятью, методы генерация параллельных итераций и организации локальных обменов данными. [1] Проводится оценка ускорения вычислений и производительности систем в рамках программ над системами типов, допускающими статический контроль. (Примеры программ: сортировка, умножение матриц, факториал.)

Рассматривается зависимость ускорения вычислений от числа процессоров и объема общей и распределенной памяти. Оценка производительности систем для высокопроизводительных вычислений учитывает влияние дисциплины работы с памятью на характеристики процессов. Рассматривается техника отладки программ, использующих защищенную и размазанную память. Изучаются решения, принятые в разных

ЯП, по работе с многоуровневой и разнородной памятью (доступ, побочный эффект, дубли и копии). Отмечается, что обработка транзакций становится одной из типовых семантик работы с памятью в ЯП. [1,3,8]

Кроме того рассматриваются средства и методы типизации управления процессами, удобными для подготовки программ, ориентированных на исполнение с помощью Open MP или MPI, а также идеи ЯП по сетевому представлению типов управления и дисциплины работы с памятью. Компонентно-ориентированная разработка ПО может следовать единому сетевому определению семантики ЯП и процесса разработки программ. [1]

5 Параллельное программирование в ЯВУ

Независимая разработка специализированных языков параллельного программирования и языков управления процессами дала ряд интересных идей по представлению и масштабированию параллельных вычислений, с которыми можно ознакомиться в материалах о языках APL, Setl, Occam, БАРС, Поляр и др. [1,4,5,10] Характерно внимание комбинаторике компонентов, адаптированных к полному пространству независимо программируемых решений. Эти идеи нашли место в ЯВУ нового поколения, таких как Sisal, Python, Haskell, C#, F# и др. [2,6,8]

Исторически первое предложение по организации языка высокого уровня для параллельных вычислений было сделано в 1962 году Айверсоном в виде языка APL [10]. Был предложен интересный механизм реализации многомерных векторов, приспособленный к расширению и распараллеливанию обработки. И в настоящее время для большинства специализированных языков параллельного программирования типично, что сложные построения факторизуются с учетом особенностей структуры данных так, что выделяются несложные отображающие функции, "просачиваемые" по структуре данных с помощью функций более высокого порядка - функционалов.

Представляет интерес эксперимент по развитию теоретико-множественной семантики языка Setl, в котором весьма общее построение формул с кванторами над множествами погружено в обычную фортрановскую схему последовательного управления процессами. Реализация языка Setl характеризуется богатым полиморфизмом. В результате программируемые функции не могут зависеть от реализационной структуры данных. В управлении процессами используется понимание команд ввода-вывода как позиций независимого порождения процессов. Такое понимание естественно согласуется с идеями теории множеств о независимости элементов множеств. [10]

Более конкретно изучаются средства организации параллельных процессов средствами языка функционального программирования F# с библиотеками .Net. На этом языке рассматривается возможность реорганизация программ. Анализируется вклад типизации данных в построение эффективных программ с использованием модифицируемой и защищенной памяти. Рассматривается стыковка программ на новых языках с производственными системами, разрабатываемыми на базе библиотек .Net. C# дает возможность встраивать функциональные построения в контекст императивных программ, а также возможность оперировать деревом разбора программы и ее исполнимым кодом. [6,8]

Рассматривается строго функциональный подход к спецификации параллельных программ и типов данных в языке Haskell, интересном благодаря предпочтению так называемой "ленивой" схемы вычислений, что дает основу для сопоставления достоинств и недостатков ленивых и энергичных методов вычислений. Уясняется концепция "монад" в строго функциональном языке программирования и особенности определения семантики языковых конструкций. Осваивается методика обеспечения удобочитаемости программ и их отладки, а также мемоизации как инструмента снижения сложности вычислений.

Изучается и язык Python, зарекомендовавший себя как удобное средство разработки распределенных систем и сетевого программирования. Рассматриваются особенности представления программ на языке Python и реализованные в нем решения по организации процессов в сетях и доступа к базам данных, история языка и особенности быстрой разработки программ.

Языки параллельного программирования занимают видное место среди функциональных языков. Довольно известен язык функционального программирования Sisal, авторы которого создали интересный прецедент по расширению и уточнению системы понятий программирования для нужд представления и реализации масштабируемых параллельных вычислений. Программа в этом языке строится из участков с однократными присваиваниями, удобными для оптимизирующих преобразований, включая распараллеливание. В структуре цикла выделены позиции для формирования пространства параллельных итераций, фильтрации или сборки параллельно полученных результатов и обработки потоков данных при развитой системе работы с векторами, включая методику распространения скалярных операций на структуры данных. [2]

Основные идеи языков параллельного программирования APL и VAL, предшественника языка Sisal, были обогащены в языке БАРС [4] в трех направлениях:

1. в качестве базовой структуры данных были выбраны комплексы, представляющие собой нечто вроде размеченных множеств с возможностью обозначить кратность вхождения элементов,
2. описание элементов памяти сопровождается предписанием дисциплины доступа к памяти,
3. средства управления асинхронными процессами включают механизм синхросетей, позволяющий согласовывать функционирование узлов из независимо представленных фрагментов.

Процедуры в таком языке приспособлены к варьированию дисциплины доступа к данным и схемы управления процессами обработки комплексов.

6 Заключение

Парадигмы параллельного программирования характеризуются сочетанием низкоуровневых средств управления процессами обработки событий (семафоры, рандеву) и высокоуровневых методов представления иерархии данных (вектора, структуры и размеченные объединения), дополненным механизмами сверхвысокого уровня, допускающими формирование сложных комплексов из независимо программируемых компонент. Эти линии выходят за пределы обычных основных курсов по программированию. Поэтому материал разрабатываемого курса затрагивает ряд дополнительных разделов университетских курсов по информатике, рекомендованных CC2001 [11], таких как:

AL11. Параллельные алгоритмы
AR9. Архитектуры сетевых и распределенных систем
OS3. Параллелизм
OS5. Управление памятью
OS11. Оценка производительности систем
PL7. Функциональное программирование
PL8. Системы трансляции
PL9. Системы типов
PL10. Семантика ЯП
PL11. Разработка ЯП
IM7. Обработка транзакций
SE9. Компонентно-ориентированная разработка ПО
CN4. Высокопроизводительные вычисления

Кроме того, имеется тематическое пересечение с углубленными курсами:

CS343. Парадигмы программирования
CS344. Функциональное программирование
CS368. Символьные вычисления
CS396. Компонентное программирование

Проблемы разработки, отладки, тестирования и верификации параллельных программ по сложности превосходят обычное программирование. Именно здесь сосредоточена исследовательская активность современного языкотворчества и системного программирования, поиск средств и методов интеграции удачного опыта параллельного программирования и успешного обучения методам параллельных вычислений. Для практического анализа и поиска подходящих решений в поддержку курса разрабатывается учебный язык параллельного программирования, содержащий подъязык начального обучения школьников, не имеющих опыта программирования.

ЛИТЕРАТУРА:

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления.- СПб.: БХВ-Петербург, 2002. - 608 с.
2. Городняя Л.В. Основы функционального программирования. - М.: Интернет-Университет Информационных технологий. - http://www.intuit.ru, 2004. - 272 с.
3. Д.В. Иртегов Многопоточное программирование с использованием POSIX Threads -http://www.intuit.ru/department/se/posixthreads/
4. Котов В.Е. МАРС: архитектуры и языки для реализации параллелизма. // Системная информатика. Вып 1. Проблемы современного программирования. - Новосибирск: Наука. Сиб. отд-ние, 1991. - с.174-194.
5. Лельчук Т.И., Марчук А.Г. Язык программирования Поляр: описание, использование, реализация. - Новосибирск, 1986. - 94 с.
6. Д.В. Сошников Функциональное программирование http://www.intuit.ru/department/pl/funcprog/lit.html (примеры на F#)
7. Хоар Ч. Взаимодействующие последовательные процессы. - М.: Мир, 1989 - 264 с.
8. C# Language Specification Version 3.0
9. Robin Milner, Communication and Concurrency, Prentice Hall, International Series in Computer Science, ISBN 0-131-15007-3. 1989
10. http://setl.org/setl/ About GNU SETL
11. Рекомендации по преподаванию информатики в университетах. Computing Curricula 2001: Computer Science. Пер. с англ. Ред. перевода: В.Л.Павлов, А.А.Терехов. - СПб.: СПбГУ, 2002. - 188 с.