ПАРАДИГМЫ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ В УНИВЕРСИТЕТСКИХ ОБРАЗОВАТЕЛЬНЫХ ПРОГРАММАХ И СПЕЦИАЛИЗАЦИИ

Л.В. Городняя

Рассматриваются парадигмы параллельного программирования от уровня базовых средств управления взаимодействующими процессами до уровня разработки программ высокопроизводительных вычислений. Дана характеристика языков параллельного программирования, изучаемых в курсе <Парадигмы программирования> [23]. Работа поддержана грантом РФФИ 08-01-00899а.

ВВЕДЕНИЕ

В середине 1970-х годов активное исследование методов параллельного программирования рассматривалось как ведущее направление преодоления кризиса технологии программирования. Рост интереса к параллельному программированию связан с переходом к массовому производству многоядерных архитектур. Роль параллелизма в современном программировании подробна рассмотрена [13,15].

Проблемы определения языков параллельного программирования связаны с дистанцией между уровнем абстрактных понятий, в которых описываются решения сложных задач, и уровнем конкретных аппаратных средств и архитектурных принципов управления параллельными вычислениями (потактовая синхронизация, совмещение действий во VLIW-архитектурах, сигналы, семафоры критических участков, рандеву и т.п.)

С параллельными процессами студенты встречаются при работе на уровне ОС, при организации практики на базе суперкомпьютеров, при сетевой обработке данных, при компиляции учебных программ и т.п. Проблемы подготовки параллельных программ для всех столь разных работ обладают общностью, но есть и существенная специфика, требующая понимания разницы в критериях оценки программ и информационных систем для различных применений.

ОПЕРАЦИОННЫЕ СИСТЕМЫ

Управление процессами на уровне ОС заключается в оперировании заданиями, сводимыми к передаче данных между устройствами и файлами [5,8,12]. Языки уровня ОС характеризуются полнотой, обеспечивающей реализацию эффективных решений по доступу к оборудованию и надежности информационной обработки долговременно хранимых данных [1, 22].

Параллельное программирование на уровне ОС выглядят как нечто среднее между программированием на макроассемблере и языках высокого уровня. Различие проявляется в понимании данных и команд:

- Роль данных выполняют файлы.
- Файлы могут участвовать одновременно в разных процессах.
- Выполнение команды рассматривается как событие.
- Событие может быть успешным или неудачным.
- Реакция на событие программируется как обработчик события.
- Программа процесса может быть нацелена на непрерывное обслуживание заданий.
- Возможна синхронизация процессов и порождение подчиненных процессов.
- Программа процесса создается как данное, которое может применяться равноправно с командами.

В результате разработка программ для организации взаимодействия процессов отличается от подготовки обычных последовательных программ на весьма глубоком уровне, что можно показать на модели интерпретирующего автомата для языка управления процессами. Такой автомат требует реализации дополнительной таблицы событий, протокола выполненных команд и структуры данных для очередей, регулирующих доступ к объектам. Обычно используется две модели - супервизор, контролирующий взаимодействие семейства процессов, или автомат, способный тиражировать себя при ветвлении процессов. Функционирование автомата сводится к бесконечному циклу анализа происходящих событий, появление которых влечет включение обработчиков, соответствующих событиям. Проблема остановки решается вне языка - на уровне базовых средств или внешним образом через прерывания.

Основная работа сводится к управлению заданиями, нацеленными на эффективную загрузку общего оборудования и других ресурсов. Доступ к общим ресурсам регулируется с помощью очередей запросов на обслуживание

на базе моно- и/или мульти-процессорных комплексов. Обслуживание носит асинхронный характер, формулируемый в терминах событий, что сближает технику программирования процессов с ООП. Основной критерий - возможность продолжить выполнение заданий без принципиальных потерь информации.

Появление информационных сервисов выплеснуло проблематику параллельных процессов на рядового пользователя. Это повышает роль такого критерия как понятность принципов сетевой обработки информации, что в значительной мере требует изучения механизмов взаимодействия процессов. Учебная демонстрация взаимодействующих процессов может быть показана на примере исполнителей <Муравей> и <Машинист> языка начального обучения программированию Робик [4].

ЯЗЫКИ ВЫСОКОГО УРОВНЯ

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.

По мнению Т. Хоара "Параллельная композиция действий внешне не сложнее последовательного сочетания строк в языке программирования" [19]. Функциональное программирование на Лиспе и других языках благодаря необычности (мало смягчившейся за 40 с лишним лет) и более гибкой модели организации вычислений позволяет предоставить простые примеры для показа непривычных идей параллелизма, интуитивное понимание которых не требует напряжения [17,20]. В основе - выделение списков, порядок вычисления элементов которых может быть произвольным.

Первая абстракция при моделировании процессов - исключение времени, т.е. отказ от ответов на вопрос происходят ли события строго одно за другим. Это обеспечивается следующими договоренностями:

- элементарные действия исполняются мгновенно,
- протяженное действие: всегда пара событий - начало и конец,
- нет точной привязки действий к моменту времени,
- определены отношения "раньше - позже", "одновременно", "независимо",
- совместность событий понимается как отношение "синхронизация",
- одно событие из независимых возникает в любом порядке, без причинно-следственной связи.

Обычно для решения конкретной задачи в терминах событий вводится специальный объект - автомат, способный реагировать на заданное множество событий в зависимости от состояния автомата. Поведение объекта, который способен действовать, описывается в виде системы правил, внешне похожих на уравнения. Решением такой системы уравнений являются процессы, которые может выполнить автомат. Выделяется начальное меню процесса для выбора из произвольного числа альтернатив. Если для всякой альтернативы меню дальнейшего поведения совпадают, то процессы тождественны. Формулировка такого рода закономерностей позволяет разделить описание процесса на уровень спецификации и реализации. Сравнивая запись меню процесса с представлением грамматики языка, можно обратить внимание на их подобие. Множества текстов или языки используются как характеристика, показывающая разнообразие вариантов хода событий - потенциал модели процессов.

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.

Функциональный подход к параллельному программированию ценен возможностью унификации понятий, различие которых мало существенно с точки зрения их реализации. При необходимости можно одинаково работать с процессами, объектами, системами и их контекстом. Все это определенные структуры данных с заданной схемой взаимодействия и разными предписанными ролями в более общем процессе информационной обработки. При необходимости взаимодействие процессов может быть пошагово синхронизовано. Взаимодействия процессов естественно могут быть заданы как взаимосвязанные рекурсивные функции.

Существование проблемы второго языка объясняет, что достаточно распространены методы представления параллельных программ, через добавление средств параллелизма в популярные языки программирования, такие как Fortran, Pascal, Ada и др. [2] Существуют версии ряда стандартных языков императивного программирования, приспособленные к выражению взаимодействия последовательных процессов в предположении, что в каждый момент времени существует лишь один процесс. При таком подходе в программе выделяются критические интервалы, учет которых полезен при распараллеливании программ [11]. Многие традиционные языки программирования при-

способлены к выражению параллелизма с помощью специальных расширений или библиотечных функций, обеспечивающих выделение участков с независимыми действиями, пригодными для распараллеливания компилятором.

ЯЗЫКИ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ

Применение параллельных архитектур повышает производительность при решении задач, явно сводимых к обработке векторов. Но 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.

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, Sisal, БАРС, Occam, Поляр и др. [1,3,8,10,12] Характерно внимание комбинаторике компонентов, адаптированных к полному пространству независимо программируемых решений.

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. Сложные данные представляются как пара из последовательности скаляров и паспорта, согласно которому эта последовательность структурируется. В результате любое определение функции над скалярами автоматически распространяется на произвольные структуры данных из однотипных скаляров.

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. В результате можно независимо варьировать структуры данных, функционалы, отображающие функции, методы сборки полного результата отображения и набор отображаемых множеств. Целенаправленно выделяются конвейерные процессы, приспособленные к минимизации хранения промежуточных результатов [1,3,10].

Языки параллельного программирования занимают видное место среди функциональных языков. Довольно известен язык функционального программирования Sisal. Название языка расшифровывется как <Streams and Iterations in a Single Assignment Language>. Sisal представляет собой результат развития языка VAL, известного в середине 70-х годов. Среди целей разработки языка Sisal следует отметить наиболее характерные, связанные с функциональным стилем программирования [3].

- Создание универсального функционального языка.
- Разработка техники оптимизации для высокоэффективных параллельных программ.
- Достижение эффективности исполнения, сравнимой с императивными языками типа Fortran и C.
- Внедрение функционального стиля программирования для больших научных программ.

Эти цели создателей языка Sisal подтверждают, что функциональные языки способствуют разработке корректных параллельных программ. Одна из причин заключается в том, что функциональные программы свободны от побочных эффектов и ошибок, зависящих от реального времени. Это существенно упрощает отладку. Результаты переносимы на разные архитектуры, операционные системы или инструментальные окружения. В отличие от императивных языков, функциональные языки уменьшают нагрузку на кодирование, в них проще анализировать информационные потоки и схемы управления.

Легко создать функциональную программу, которая является безусловно параллельной, если ее можно писать, освободившись от большинства сложностей, связанных с выражением частичных отношений порядка между отдельными операциями уровня аппаратуры. Пользователь Sisal-а получает возможность сконцентрироваться на конструировании алгоритмов и разработке программ в терминах крупноблочных и регулярно организованных построений, опираясь на естественный параллелизм уровня постановки задачи.

Как и свойственно языкам функционального программирования, язык Sisal математически правилен - функции отображают аргументы в результаты без побочных эффектов, и программа строится как выражение, вырабатывающее значение. Форма параллельного цикла включает в себя три части: генератор пространства итераций, тело цикла и формирователь результата. Такая форма, дополненная требованием однократности присваиваний в любом блоке, хорошо приспособлена к распараллеливанию при компиляции.

Sisal-программа представляет собой набор функций, допускающих частичное применение, т.е. вычисление при неполном наборе аргументов. В таком случае по исходному определению функции строятся его проекции, зависящие от остальных аргументов, что позволяет оперативно использовать эффекты смешанных вычислений и опре-

делять специальные оптимизации программ, связанные с разнообразием используемых конструкций и реализационных вариантов параллельных вычислений.

Основные идеи языков параллельного программирования APL и VAL, предшественника языка Sisal, были обогащены в языке БАРС [1] 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 включали 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. Синхросети позволяют независимые описания процессов связывать в терминах разметки. Узлы с одинаковой разметкой срабатывают одновременно. Полное представление об асинхронных процессах, их эффективности и проблемах организации дают работы по сетям Петри [9].

Перспективно применение универсальных языков сверх высокого уровня, таких как Setl, Python, Cw и т.п., абстрагирование данных и процессов в которых приспособлено к гибкому и строгому структурированию, удобному для доказательных построений [6,7,16,21]. В этом плане 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. Для представления множеств используется около двадцати разных структур данных, выбор которых осуществляется системой программирования в зависимости от динамики операций над множествами. В результате программируемые функции не могут зависеть от реализационной структуры данных.

В практике управления процессами используется понимание команд как позиций независимого порождения процессов. Такое понимание естественно согласуется с идеями теории множеств о независимости элементов множеств. Не исключено, что принятая в языке Sisal схема управления процессами может успешно заменить фортрановский стиль, принятый в языке Setl.

ЗАКЛЮЧЕНИЕ

В целом, парадигмы параллельного программирования можно характеризовать сочетанием низкоуровневых средств управления процессами обработки событий (семафоры, рандеву) и высокоуровневых методов представления иерархии данных (вектора, структуры и размеченные объединения) с просачиваемыми по такой иерархии действиями в стиле объектно-ориентированного программирования, дополненным механизмами сверхвысокого уровня, допускающих или обеспечивающих формирование сложных комплексов из независимо программируемых компонент.

В работах ИСИ СО РАН им. А.П. Ершова по исследованию и классификации компьютерных языков, нацеленых на реализацию идей А.П. Ершова и С.С. Лаврова о создании лексикона программирования, заметное место занимает исследование парадигм программирования и особо - парадигм параллельного программирования, обладающих большим разнообразием вычислительных моделей и подходов к аппаратной поддержке высокопроизводительных вычислений.

ЛИТЕРАТУРА:

1. А.В. Быстров, Н.Н. Дудоров, В.Е. Котов О базовом языке. - В сб.: Языки и системы программирования. - Новосибирск, 1979, с. 85-106.
2. П. Вегнер Програмирование на языке Ада - М.: Мир, 1983, 240 с.
3. В.А. Евстигнеев, Л.В. Городняя, Ю.В. Густокашина "Язык функционального программирования SISAL", // Интеллектуализация и качество программного обеспечения, Новосибирск, 1994, с. 21-42
4. Г.А. Звенигородский Язык начального обучения Робик в учебной системе программирования. - Новосибирск, 1982, Сб. <Программное обеспечение задач информатики>, с. 72-85
5. А. Колин Введение в операционные системы. - М.: МИР, 1975. - 116 с.
6. Д.Я. Левин Язык сверхвысокого уровня СЕТЛ - Новосибирск. <Наука>, 1983. 160 с.
7. И. Лейнингем Освой самостоятельно Python. - М. <Вильямс>, 444 с.
8. Т.И. Лельчук, А.Г. Марчук Язык программирования Поляр: описание, использование, реализация. - Новосибирск, 1986. - 94 с.
9. И.А. Ломазова Вложенные сети Петри. - М. <Научный мир>, 207 с.
10. Н.А. Магариу Язык программирования АПЛ. - М. <Радио и связь>, 96 с.
11. Пересмотренное сообщение об Алголе-68. - М.: Мир, 1979, 533 с.
12. Руководство по языку Оккам. - Новосибирск, 1987. - 75 с.

13. Т. Пратт, М. Зелковиц Языки программирования. Разработка и реализация. - М. <Питер>, 2002, 688 с.
14. Л. Прехельт Эмпирическое сравнение семи языков программирования. - М.: Открытые системы, 12(56), 2000. - С. 45-52.
15. Э. Танебаум, М. Ван Стеен Распределенные системы. Принципы и парадигмы. - М.: "Питер", 876 с.
16. Д. Уоткинс, М. Хаммонд, Б. Эйбрамз - Программирование на платформе .Net. М. <Вильямс>, 2003. - С. 367
17. П. Хендерсон Функциональное программирование. - М.: Мир, 1983
18. Б. Хигман Сравнительное изучение языков программирования. - М.: Мир, 1974. - 204 с.
19. Ч. Хоар Взаимодействующие последовательные процессы. - М.: <Мир>, 1989. 264 с.
20. P. Graham ANSI Common Lisp. //Prentice Hall,1996, 432p
21. H.S. Warren, Jr. ASL: A Proposed Variant of SETL. - New York University, 1973, 326 p.
22. Р. Петерсон. LINUX. Руководство по операционной системе. - К. BHV, 1997. - 688 с.
23. Городняя Л.В. Парадигмы программирования. М: Интернет-университет информационных технологий. http://www.intuit.ru, 2007.
24. А.Г. Марчук, Л.В. Городняя, Ф.А. Мурзин, Н.В. Шилов. Классификация компьютерных языков: состояние, проблемы перспективы. - СПб, Материалы всероссийской конференции "Космос, астрономия, программирование (Лавровские чтения)", 20-22 мая 2008
25. Т.А. Андреева, И.С. Ануреев, Е.В. Бодин, Л.В. Городняя, А.Г. Марчук, Ф.А. Мурзин, Н.В. Шилов. Компьютерные языки как форма и средство представления, порождения и анализа научных и профессиональных знаний. - СПб, Всероссийская конференция "Телематика", 2008, http://tm.ifmo.ru/tm2008/db/doc/get_thes.php?id=214