О проблеме начального обучения параллельному программированию

Л.В. Городняя
Институт систем информатики им. А.П. Ершова СО РАН,
Новосибирск

Обсуждается проблема обучения параллельному программированию. Дана характеристика языков программирования, изучаемых в курсе «Парадигмы программирования», для студентов ФИТ и ММФ НГУ. На этапе перехода к многопроцессорным архитектурам возрастает актуальность обучения параллельному программированию, что требует развития языково-информационной поддержки введения в программирование. Предлагается проект языка начального обучения параллельному программированию. Работа поддержана грантом РФФИ 08-01-00899-а.

Введение

Мир программирования и его техническая основа претерпели значительные изменения за последние двадцать лет. Возрастает актуальность обучения параллельному программированию, что требует новой языково-информационной поддержки введения в программирование [1]. Для систематичной подготовки специалистов в области параллельного программирования необходима гибкая инструментальная поддержка учебной практики, поддерживающая разные модели параллелизма. Однако для многих языков высокого уровня (ЯВУ) характерно включение в семантику языка одной модели параллелизма. Поэтому становится актуальной проблема создания языка начального обучения параллельному программированию (ЯНОП), предоставляющего основные модели параллелизма для обучения результативному представлению естественного параллелизма при решении важных задач.

При разработке ЯНОП рассмотрен ряд моделей параллелизма, предложенных в языках программирования Apl, Algol-68, Ada, Setl, Sisal, F# и др. Проанализированы требования к языковой и инструментальной поддержке процесса обучения параллельному программированию. Проект учебного языка параллельного программирования нацелен на ознакомление с основными моделями параллелизма, встречающимися в учебно-методических и научных материалах [2], языках высокого уровня, языках высокопроизводительного программирования и при организации сетевых информационных сервисов.

Определены уровни и цели начального (полгода в 10–12 лет) и предпроизводственного (2 года с/к для старшеклассников и студентов) обучения в контексте истории языков программирования. Проанализированы текущие задачи разработки учебных языков программирования, дополняющих основные парадигмы программирования на ЯВУ (императивное, функциональное, логическое, объектно-ориентированное) средствами параллельного программирования (GPU, MPI, Open MP).

Сформулированы требования к языкам начального обучения параллельному программированию:

- Концентричность или разложение понятий по уровням обучения.
- Представление и преобразование моделей управления процессами.
- Определение видов памяти с разной дисциплиной доступа.
- Настраивание на реалии производственных языков и систем.

Выбор решений

Многие затруднения с параллельным программированием имеют образовательный характер. Концепции взаимодействующих процессов следовало бы осваивать при начальном знакомстве с информационными технологиями и программированием.

При проектировании ЯНОП учтены факторы успеха наиболее известных учебных языков программирования, таких как Basic, Pascal, Logo, Grow, Робик и др., с целью проявления их специфики и типовых свойств. Ведущее понятие языка начального обучения программированию Робик – «исполнитель», причем исполнителей может быть много, и они могут обладать разными системами команд. Такой подход в ЯНОП допускает естественное развитие представления программ в направлении моделирования асинхронных процессов.

Программы в ЯНОП строятся из потоков, а потоки – из действий, выполнение которых может быть обусловлено ожиданием времени (пауза) или другого сигнала, предикатом или вероятностью срабатывания. Последнее означает, что при исполнении ведется учет частоты выполнения вероятностных действий, как в системах для разработки компьютерных игр. Кроме того, можно объявить планируемую длительность выполнения действий.

При организации сложных данных и процессов используются общие структуры или средства композиции, такие как списки, множества

и варианты, обеспечивающие представление отношений «после», «одновременно» и «взаимоисключение».

Представление циклов на ЯНОП наследует решения функционального языка параллельного программирования Sisal, а именно, выделяются участки для формирования пространства распараллеливаемых итераций и для свертки полученных параллельно значений в общий результат.

Проект учебного языка параллельного программирования

Реализация ЯНОП образует три уровня – элементарно-учебное ядро, технолого-прагматическое расширение и вводно-ознакомительную оболочку.

Ядро реализации ЯНОП составляют базовые средства уровня операционной системы (ОС) – условия готовности и завершения процесса, очереди и серийная обработка, типичные для ОС UNIX и языка Java. Такие средства рассматриваются как основа для реализации схем управления, встречающихся в ЯВУ (критические участки и семафоры, рандеву, защищенные команды, конвейеры), навыки программирования и применения которых входят в задачу элементарно-учебного уровня.

Остальное формируется как расширения ядра языка, выполняемые с целью достижения понимания реализационной семантики параллельного программирования для современных средств, что и является задачей технолого-прагматического уровня, обеспечивающего решение учебных задач, приведенных в [1,2]. Отдельные приемы программирования на ЯНОП показаны ниже.

Пример 1. Программа масштабируемого решения олимпиадной задачи для младших школьников: «Как поджарить 3 котлеты на 2 сковородах за минимальное время?». Типовое решение: «Поджарив первую сторону двух котлет, одну котлету отложить на тарелку, а на освободившуюся сковороду положить третью котлету и т.д.». Здесь после жарки одной стороны освобождаются обе сковороды.

```
ОЧЕРЕДЬ жарить = (2 2 2),%% Предстоит обжарить 2 стороны 3 котлет
          готово = ( ) ;   %% Дана пустая тарелка для готовых котлет
ВЕКТОР сковороды [1..2] ;  %% Имеется 2 сковороды для жарки

  ПОКА жарить ПОВТОР     %%  цикл работает до опустошения очереди

    жарить → сковороды  ;  %%  в вектор из очереди переданы два элемента
                           %%  переданные элементы исчезают из очереди
```

```
ЖДУ 5;                                  %% Время «жарки»
сковороды = (сковороды - 1) ;           %% Одна сторона обжарена

%% Разделяем котлеты на  готовые и требующие жарки
[ (жарить ; (сковороды ≠ 0) ), %% «Недожаренные» идут в очередь на жарку
(готово ; (сковороды = 0) ) ] ;     %% «Готовые»  -  в тарелку (со 2-го витка)

РЕЗУЛЬТАТ  готово ;                     %% = (0 0 0 )
```

Сравнения *(сковороды ≠ 0)* и *(сковороды = 0)* выполняют роль фильтров: выделяют элементы, удовлетворяющие сравнению. Выражение **(**x **;** y**)** – пополнение очереди X элементами из Y.

Пример 2. Варианты программы «**Простой торговый автомат**» [2].

!x – прием X, **? x –** выдача X.

Последовательно: *прием монетки,* *потом выдача шоколадки*	Асинхронно: *Прием монетки и выдача шоколадки* *в любом порядке*
ПОВТОР **(!** мон **; ?** шок **)**	ПОВТОР **[!** мон **, ?** шок **]**

Пример 3. Расстановка барьеров для взаимодействия процессов

```
ПТА =                                   УЧЕТ =
   ПОВТОР                               шоколадки = N
      ( ! мон ; ? шок ;                    ПОКА шоколадки ПОВТОР
      Контроль:          %% барьер            Контроль:          %% барьер
         ИНОГДА%  ? сувенир                   шоколадки = шоколадки – 1
%% иногда выдается «приманка»          СТОП
)

ПТА_С_УЧЕТОМ = ( Контроль ) [ ПТА , УЧЕТ ]  % общий барьер в потоках
```

Пример 4. Переименование барьера и подстановка действий

```
УЧЕТ_Б = (  N  b )          ПОКА N  ПОВТОР   b:  N = N – 1

Ф_ПТА_Б =  (мон шок p)        ( ! мон ; ? шок ;   p:  Ф_ПТА_Б )

    (Контроль) [ Ф_ПТА_Б ( жетон, игрушка, Контроль ) ,
                                УЧЕТ ( 4, Контроль )]   % в автомате 4
игрушки
```

Литература

1. **Воеводин В.В., Воеводин Вл.В.** Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.
2. **Хоар Ч.** Взаимодействующие последовательные процессы. М.: Мир, 1989. 264 с.
3. **Робик** [Электронный ресурс] / Википедия. Свободная энциклопедия. 2009 Режим доступа: http://en.wikipedia.org/wiki/Cloud_computing, свободный.