УДК 004.43

ОПИСАНИЕ ПАРАЛЛЕЛИЗМА В ФУНКЦИОНАЛЬНОМ ЯЗЫКЕ ПОТОКОВО-ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ

Матковский И.В.
научный руководитель д-р техн. наук Легалов А. И.
Сибирский Федеральный Университет

Огромное разнообразие различных параллельных вычислительных систем (ПВС) является естественными результатом популяризации параллельных вычислений; к сожалению, есть у этого разнообразия и свои недостатки. В большинстве случаев для каждой ПВС существует собственный подход к параллельному программированию; игнорирование особенностей конкретной системы сводит на нет все её преимущества. Этот фактор сильно затрудняет разработку кроссплатформенных приложений – перенос некоей программы с одной ПВС на другую очень часто сопровождается радикальной переработкой этой программы. Нередко перерабатывать программу приходится даже в тех случаях, когда исполняющая её ПВС не меняет свой тип, но изменяет конфигурацию – даже изменение скорость работы каналов передачи данных способно оказать большое влияние на эффективность используемой схемы параллелизма. Как следствие, актуальной является проблема создания метода написания легко переносимых с одной ПВС на другую приложений.

Одним из таких методов является архитектурно-независимое программирование; в рамках этого подхода предполагается, что программа разрабатывается для системы, поддерживающей неограниченный параллелизм и неограниченный объем вычислительных ресурсов. Разработанная и отлаженная с учетом данных предпосылок программа может быть исполнена на конкретной ПВС с помощью отдельного модуля, который будет осведомлен о возможностях и ресурсах данной системы и применит их к программе. Отделение разработки программы от её привязки к конкретной ПВС упрощает работу программиста.

Целый ряд довольно популярных подходов к архитектурно-независимому программированию построен на использовании функциональных языков; эффективным оказалось представление программы в виде реверсивного информационного графа (РИГ), описывающим потоки по данным. Управление вычислениями (и управляющие потоки) в таких случаях задаются в виде некой внешней сущности – управляющего графа (УГ).

Пифагор представляет собой язык для написания функционально-потоковых параллельных программ. Отсутствие привязки к конкретной вычислительной системе и учета ресурсов позволяет создавать программы, описывающие максимальный параллелизм.

Реализуется параллелизм в языке “Пифагор” на основе так называемых параллельных списков. Параллельные списки являются не составными объектами, но группами независимо существующих друг от друга отдельных объектов. Объединяться в параллельные списки могут как функции, так и выступающие в качестве их аргументов данные.

Передача параллельного списка A в функцию F в качестве аргумента равнозначно одновременному вызову функции F с каждым из элементов списка A в качестве аргумента; результаты вызовов функции будут помещены в новый параллельный список. Так, результатом вычисления выражения [1,2,3,4]:f станет [1:f, 2:f, 3:f, 4:f].

Передача одного аргумента X в параллельный список A из нескольких функций равнозначна одновременному вызову входящих в список A функций с аргументом X;

результаты вызовов функций будут помещены в новый параллельный список. Так, результатом вычисления выражения 1:[f1,f2,f3,f4] станет [1:f1, 1:f2, 1:f3, 1:f4].

Два описанных выше варианта фактически являются частными случаями третьего. Передача параллельного списка аргументов A в параллельный список функций F равнозначна одновременному вызову всех существующих комбинаций (аргумент:функция). Результаты вызовов помещаются в новый параллельный список. Так, результатом вычисления выражений [A1,A2,A3]:[F1,F2] станет [A1:F1, A1:F2, A2:F1, A2:F2, , A3:F1, A3:F2].

Рассмотрим интерпретацию параллельного списка аргументов параллельного списка функций на примере. Изначально оба параллельных списка пусты (рисунок 1а). Пусть первыми поступают аргументы A1 и A2 (рисунок 1б); на данном этапе ни для одной из существующих операций интерпретации данных еще не достаточно. Пусть далее в список поступает аргумент A3 (рисунок 1в); это также операцию интерпретации не запустит. Пусть после этого в список функций поступит функция F1; это даст возможность запустить операции A1:F1, A2:F1, A3:F1 (рисунок 1г). Пусть после этого в список функций поступит F2; её также необходимо применить к каждому из имеющихся аргументов – проделав A1:F2, A2:F2, A3:F2 (рисунок 1д). Пусть после этого в список аргументов поступит аргумент A4; применим к нему все имеющиеся на данный момент функции – F1 и F2 (операции A4:F1 и A4:F2) (рисунок 1е).

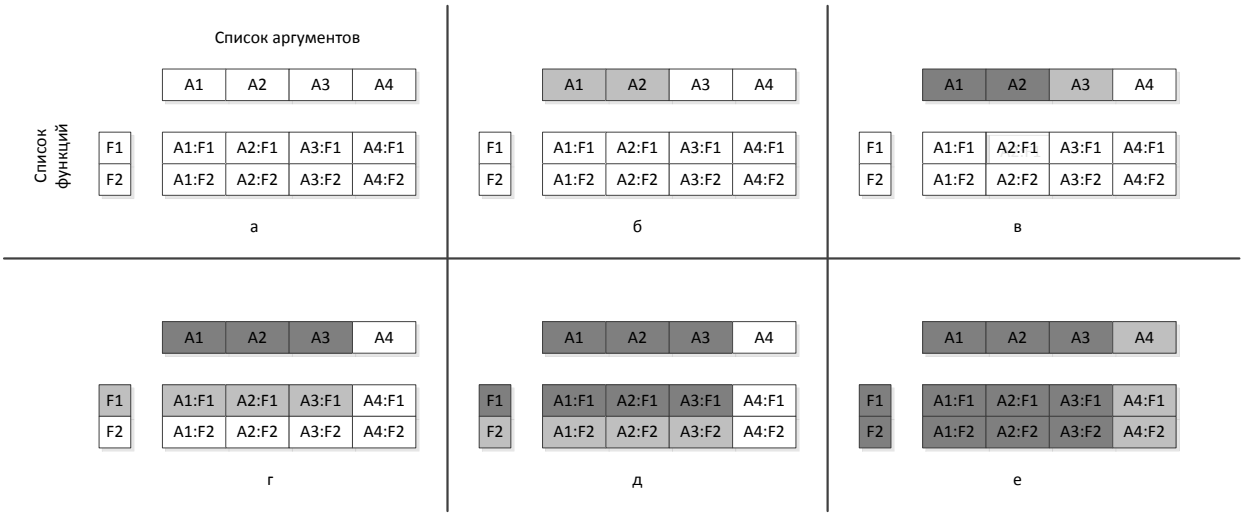

Рисунок 1 – Обработка параллельных списков

Параллельные списки позволяют описывать максимальный, не зависящий от ресурсов конкретной ПВС параллелизм; наложение соответствующих системе ограничений лежит целиком и полностью на интерпретирующей системе.