

О РОЛИ ОБЪЕМНЫХ ПРОГРАММ ПРИ ОБУЧЕНИИ МНОГОПОТОЧНОМУ ПРОГРАММИРОВАНИЮ

С.Е. Гвоздев, А.П. Калинина, Н.А. Калинина

Появление доступных широкому пользователю многоядерных архитектур сделало актуальным подготовку соответствующих специалистов. С одной стороны, возникла необходимость дополнения базовых курсов информатики элементами многопоточного программирования.

Попытка подобной модернизации курсов программирования была проведена на первом курсе НГУ в рамках курса «Программирование» на механико-математическом факультете. Вначале применяемая методика была апробирована в рамках спецкурса «Практическое параллельное программирование в системах с общей памятью». Необходимым требованием предварительных знаний было наличие представлений о языке программирования C на простейшем уровне.

Достаточно успешным оказалось внедрение тренинговой методики, применяемой обычно большинством компьютерных фирм для обучения собственных сотрудников. Суть этой методики представляет собой поэтапный разбор простых примеров, каждый из которых является последовательностью программных версий кода. Каждая последующая версия была незначительной модификацией предыдущей в сторону более простого введения параллелизма.

Например, обучение конструкциям явного параллелизма происходило следующим образом. В частности, вначале предлагалось распараллеливаемый код данной задачи выделить в отдельную функцию. Из главной программы выполнялось однократное обращение к данной функции. Следующим шагом было введение концепции нескольких последовательных исполнителей. Обращение к данной функции из данной программы выполнялось уже несколько раз – число обращений равнялось будущему количеству потоков. Код функции также модифицировался – у нее появлялся новый формальный параметр, равный «номеру» исполнителя, а выполняемые вычисления зависели от «номера» этого исполнителя. Но это все еще был последовательный код. Окончательным штрихом являлась «оснастка» программы параллельными конструкциями выбранной параллельной технологии. После описанной выше предварительной подготовки это было совершенно нетрудно.

Дополнение этой методики работой с отладчиками по поиску многопоточных ошибок и профилированию кода (использовались Intel Thread Checker и Intel Thread Profiler) сделало пригодной эту методику для широкого класса пользователей.

В результате применения данной методики достаточно легко снимался психологический барьер перед многопоточностью, происходило привыкание к понятиям. Однако, после изучения подобного вводного курса человек оказывался достаточно беспомощным перед объемными программами, несмотря на то, что в них реализовался практически тот же самый явный параллельный алгоритм, уже разобранный в простейших примерах.

В связи с этим наиболее логичной структурой вводного курса многопоточного программирования представлялась следующая: первая половина – психологическое привыкание и знакомство с основными понятиями на основе тренинговой методики, а вторая половина – выполнение распараллеливания достаточно объемного последовательного кода в режиме курсовой работы.

Можно выделить основные источники проблем объемного кода. Наиболее очевидным является резкое возрастание количества переменных, в которых потенциально возможны конфликты памяти. Следующей причиной была необходимость применить тот же самый алгоритм несколько раз, что также повышало вероятность ошибки. Кроме того, конкретика задачи требует знания специальных алгоритмов для разделения области видимости данных для каждого потока, например, элементов массивов. Все это, вместе взятое, должно было привести обучаемого к очевидной мысли – необходимости выработки культуры многопоточного программирования. Культура многопоточного программирования, аналогично культуре любого другого, представляет собой набор применяемых приемов и «табу».

В качестве объемной программы рассматривался TVD расчет двумерного поля течения идеального газа на основе уравнений Эйлера в канале переменного сечения. Двумерность расчета предполагает двойное применение параллельного алгоритма, а длинная цепочка функций, каждая из которых обращается к последующей, требует скупулезного разделения областей видимости всех данных. В то же время применение граничных условий разного вида на различных границах требует применения для каждого рода массивов своей функции раздвижки массивов для последующей раздачи кусков массивов потокам.

Эксперименты показали, что обучаемые, начав с попытки улучшения самой последовательной программы, быстро осознали необходимость первичной отладки параллельных конструкций и функций пересчета номеров в

массивах на простых тестах. Второе, что было быстро осознано, была необходимость повышения степени инкапсуляции кода в функции для уменьшения пространства имен переменных, в которых нужно однозначно определять область видимости данных потоками. При этом достаточно быстро пошло самостоятельное освоение отладчиков с применением их более сложных функций. Кроме того, после очистки программы от ошибок – конфликтов памяти были найдены алгоритмические ошибки в последовательном коде, что, в свою очередь, способствовало изучению неизвестного сначала численного алгоритма. Следует отметить, что без применения отладчика по поиску многопоточных ошибок работа обучаемого практически останавливалась, и лишь с помощью отладчика сдвигалась с мертвой точки. Таким образом, необходимость многократного применения «рецептов» в объемной программе успешно способствовало закреплению навыков.

Все это, вместе взятое, показало эффективность включения распараллеливания объемных последовательных программ в курсы многопоточного программирования. С этой точки зрения, TVD расчет представляется достаточно подходящим для этой цели: простота параллельного алгоритма в сочетании с объемом кода позволяет закрепить приобретенные навыки и выработать культуру многопоточного программирования. Можно отметить, что только после выполнения «курсовой» у обучаемых появлялась так необходимая вера в собственные способности, более глубокое понимание многопоточности. После же прохождения тренинговой части ощущения обучаемых можно было выразить словами: «в первом приближении понятно, но непонятно, смогу ли я это сам применить». Таким образом, завершение вводного курса многопоточного программирования «курсовой» по распараллеливанию объемного последовательного кода представляется необходимым условием успешного результата обучения.