Особенности разработки и преобразования функционально-потоковых параллельных программ *

А.И. Легалов[1], М.С. Ушакова[1]

Сибирский федеральный университет[1]

В работе рассматривается подход, который заключается в построении параллельных алгоритмов и программ, обладающих неограниченным параллелизмом. Предполагается, что в ходе последующих трансформаций будут получены альтернативные варианты наложением различных ограничений и проведением эквивалентных преобразований. Ограничения могут носить как ресурсный, так и функциональный характер, что обуславливается спецификой целевой задачи и вычислительных ресурсов. Эквивалентные преобразования позволяют заменять одни формы представления параллелизма другими. Это позволяет по-иному взглянуть не только на процесс разработки параллельных алгоритмов, но и на их анализ и трансформацию.

Разработка программ, опирающаяся на неограниченный параллелизм, чем-то напоминает нисходящее программирование. Но вместо постепенной детализации разрабатываемого алгоритма происходит аналогичное уточнение накладываемых на программу ограничений в соответствии со спецификой ее дальнейшего использования. В перспективе подобные преобразования можно проводить формально, опираясь на базу знаний и язык, допускающий трансформации исходной программы в соответствии с заложенными в него аксиомами и алгеброй преобразований, а также заменой эквивалентных по результатам функций, обладающих разным уровнем параллелизма. Несмотря на определенную оторванность от реальных вычислений, данный подход позволяет в ряде случаев получать интересные решения, так как, в отличие от написания последовательных программ, допускает использование дополнительных абстракций обеспечивающих различные варианты представления параллельных вычислений. Помимо этого переход к реальным параллельным программам может осуществляться проще за счет использования различных способов сжатия параллелизма.

В проводимых работах предлагаемый подход исследуется применительно к функционально-потоковой парадигме параллельного программирования [1]. Анализируется ряд вариантов наложения ограничений на параллелизм, а также рассматриваются способы замены одних программо-формирующих операторов другими, что позволяет проводить модификации функционально-потоковой параллельной программы с учетом особенностей предметной области или архитектуры вычислительной системы. В отличие от ранее проведенных исследований делается попытка обобщить различные методы описания параллелизма и его эквивалентных преобразований за счет использования функций высшего порядка с последующей реализацией инструментальных средств, поддерживающих проведение этих преобразований.

Можно выделить ряд способов, обеспечивающих первоначальное построение функционально-потоковых параллельных программ с последующей их трансформацией в формы, ограничивающие параллелизм:

- прямое преобразование программ в эквивалентные формы с ограниченным параллелизмом;
- использование операторов описания параллелизма, учитывающих особенности предметной области.

Прямое преобразование программ связано с наложением различных ограничений на исходный алгоритм [2], задаваемых как с учетом целевой вычислительной системы, так и на основе некоторых логических умозаключений, напрямую не связанных с конкретной архитектурой, например, определяемых оптимизацией определенных характеристик параллельного алгоритма или организацией обрабатываемых им данных. Последнее позволяет получать новые архитектурно-независимые алгоритмы, эквивалентные исходному решению, но при этом в большей степени учитывающие особенности условия реальных вычислений. При использовании языка программирования существует возможность формализованного вывода разнообразных

* Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 17-07-00288.

алгоритмов решения одной и той же задачи, которые раньше разрабатывались с использованием эвристических приемов, на основе функций высших порядков. Эти алгоритмы в дальнейшем можно использовать в качестве переносимых образцов, адаптируемых под особенности конкретной вычислительной архитектуры.

Наложение ограничений позволяет создавать не только параллельные, но и последовательные программы. Помимо этого эквивалентные преобразования параллелизма можно увязывать с применением методов формальной верификации, что обеспечивает дополнительные возможности по повышению надежности. Применение функций высшего порядка позволяет в данном случае сформировать каркасные элементы, для которых предварительно доказана эквивалентность взаимных подстановок.

Одним из факторов, определяющих поведение параллельной программы, является особенность организации и поступления данных, являющаяся отличительной чертой в разных предметных областях. В зависимости от этого зачастую определяются механизмы, описывающие параллельные вычисления. Традиционно при параллельном программировании для прикладных задач использует определенный набор базовых примитивов ориентированных на универсальную организацию вычислений. Он обеспечивает разнообразие методов управления параллельными вычислениями при использовании низкоуровневого программирования, что не вполне удобно при разработке прикладных программ. Например, в языках программирования и библиотеках ориентированных на задачи, связанные с математическими расчетами, используется более высокоуровневый набор примитивов, который не учитывает все разнообразие методов организации параллельных вычислений. Это ведет к ограничению на варианты взаимодействия параллельных фрагментов как с точки зрения уровня параллелизма, так и используемых данных. При использовании более сложных асинхронных взаимодействий часто приходится переходить на более низкий уровень программирования с применением соответствующих системных библиотек.

Организация асинхронного поступления данных также может быть представлена в виде отдельных конструкций, которые можно исследовать в экспериментальных языках программирования. Для этого в функционально-потоковую модель параллельных вычислений введен асинхронный список [3]. Его особенностью является упорядочение данных в соответствии с последовательностью их порождения. Использование асинхронных списков позволяет реализовать новый класс алгоритмов, параллелизм которых зависит от временных соотношений между выполняемыми основными и подготовительными операциями. Это повышает гибкость разрабатываемых алгоритмов, и позволяет использовать один и тот же алгоритм для динамического задания различных уровней параллелизма, однако требует дальнейших исследований.

Разработка и анализ алгоритмов, описывающих неограниченный параллелизм, предоставляет в распоряжение разработчика дополнительные методы для построения алгоритмов и программ, на организацию параллельных вычислений в которых наложены те или иные ограничения. Применение для создания таких алгоритмов языков программирования позволяет провести реальную верификацию и отладку программ без учета специфики вычислительных ресурсов, акцентируя внимание на информационной структуре. Помимо этого использование языка программирования вместо неформального описания позволяет применять инструментальные средства, обеспечивающие преобразования в автоматическом или интерактивном режиме, что может уменьшить количество ошибок, вносимых при написании параллельных программ.

Литература

1. А.И. Легалов. Функциональный язык для создания архитектурно-независимых параллельных программ. Вычислительные технологии, № 1 (10) - 2005. - С. 71-89.
2. Легалов А.И. Построение параллельных алгоритмов. – Открытые системы, № 9 (101), 2004. С. 64-68.
3. Легалов А.И. Использование асинхронных вычислений в функциональных языках параллельного программирования. – Распределенные и кластерные вычисления. Избранные материалы четвертой школы-семинара. / Институт вычислительного моделирования СО РАН. Красноярск, 2005. С. 172-183.