Методы отладки функционально-потоковых параллельных программ

Ю. В. Удалова
Сибирский федеральный университет, Красноярск

Разработка программного обеспечения немыслима без отладки и тестирования. В настоящее время эти этапы разработки программ хорошо проработаны для последовательного программирования. Существуют разнообразные отладчики, обеспечивающие пошаговый анализ кода и отображающие состояние переменных. Они также обеспечивают использование разнообразных приемов, позволяющих пропускать уже отлаженные фрагменты программ и акцентировать внимание только на требуемых функциях или процедурах.

Перенос методов отладки последовательных программ на параллельные вычисления сопровождается определенными трудностями, обуславливаемыми спецификой организации параллельных процессов. В частности, необходимы дополнительные функции, позволяющие анализировать конфликтные и тупиковые ситуации. Но в настоящее время методы отладки параллельных программ развиваются. Достигнуты определенные успехи в разработке соответствующих средств для систем, использующих потоки или процессы, расширяющие технику последовательного выполнения программ.

Существуют разнообразные способы построения параллельных программ, ряд которых существенно отличается от широко используемых в настоящее время. Их исследование может привести к перспективным технологиям, повышающим эффективность процесса разработки и использования параллельных программ. К таким способам построения следует отнести и функционально-потоковое параллельное программирование, базирующееся на модели вычислений, обеспечивающей описание ресурсно-независимого параллелизма. На основе этой модели разработан язык программирования Пифагор, позволяющий писать архитектурно-независимые параллельные программы.

Методы отладки функционально-потоковых параллельных программ требуют использования специфических приемов, что

обуславливается спецификой модели вычислений, в которой отсутствует понятие переменной. Выполняемые функции связаны напрямую, а существующие типы данных требуют нетривиальных вычислений и методов отображения результатов.

Целью работы является исследование и систематизация методов отладки функционально-потоковых параллельных программ. На данный момент разработаны три режима отладки, использующие информационный граф функционально-потоковой программы, также реализованы различные способы отображения процесса отладки.

Режимы проведения отладки

Отладка функционально-потоковой параллельной программы может проходить в одном из трех режимов: пошаговой отладке, отладке слоев и отладке ветвей.

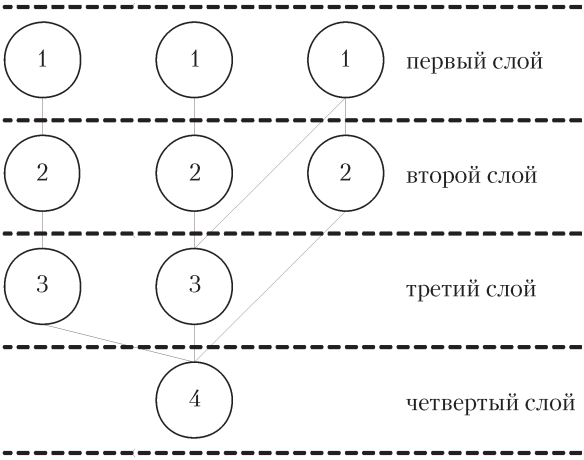


Рис. 1. Пример выделения слоев в программном графе

Режим пошаговой отладки потоковой параллельной программы, подобен режиму отладки последовательной. Возможность использования такого режима обусловлена моделью вычислений функционально-потоковой параллельной программы, определяющей программу как информационный граф, в котором вершины являются операторами, а дуги определяют связи между операторами. Параллельное выполнение основано на том, что од-

новременно выполняться могут только те операторы, между которыми нет информационных связей. Чтобы оператор мог быть вычислен, необходимо выполнение всех тех операторов-вершин, которые образуют путь к данному оператору. Отсюда следует, что непосредственное вычисление программы может пойти по другому пути, чем пошаговая отладка. Но при этом, такая ситуация не приведет к возникновению конфликтов, таких, как появление невычисленных вершин, бесконечное ожидание вычисления предыдущего результата, возникновение данных, отличных от значений, полученных при выполнении программы. Например, если выполняется множество независимых операторов, то не важно в каком порядке они были вычислены, процесс отладки все равно показывает верный результат для каждого оператора, и позволяет выявить ошибку. Если последовательность выполнения операторов идет в глубину графа, каждая отдельная ветвь может быть выполнена отладчиком только в определенном порядке, и различные ветви, не обладающие общим путем, не могут содержать взаимосвязанные операторы. Благодаря этому, переход пошагового отладчика от выполнения одной ветви к другой не приводит к противоречиям, на какой бы глубине они не произошли.

Режим отладки слоев (ярусов) предполагает, что на первом шаге отладки выполняются все те программные операторы, которые не требуют никаких входных данных, кроме входного параметра функции — первый слой. После того, как эти операторы были вычислены, в программном графе соответственные вершины помечаются как выполненные, и на следующем шаге для вычисления выбираются все те операторы-вершины, которые имеют на своем входе только вычисленные данные. Таким образом, формируется следующий слой.

Отладка по слоям предоставляет пользователю возможность наглядно увидеть количество слоев программы и количество операторов в каждом слое, т. е. фактически оценить уровень параллелизма написанной программы. Кроме того, такая схема выполнения программы при отладке наиболее точно показывает работу модели вычислений функционально-параллельных программ.

В режиме отладки слоев за один шаг отладки выполняются несколько операторов, и здесь возникает вопрос о поведении

системы при получении ошибки на одном или нескольких операторах. Такая ситуация не является критической вследствие того, что операторы одного слоя независимы между собой. Но группа операторов в следующем слое получает ошибку как входной параметр. В функционально-потоковой программе, если некоторый программный оператор получает на входе ошибку, то он все равно производит над ней необходимые действия. Чаще всего, в этом случае формируются выходные данные, также содержащие ошибку, и такие выходные данные обрабатываются последующими операторами. В процессе отладки можно рассмотреть все последствия распространения возникшей ошибки.

Режим отладки слоев обладает следующей спецификой, если среди совокупности операторов существуют вызовы функций, то такие вызовы выполнятся сразу в полном объеме, и лишь по завершении вычисления всего слоя, будет произведена отладка вызванной функции. Такая схема работы, с одной стороны, не разрушает целостности восприятия слоев, с другой стороны, делает затруднительным, но не невозможным поиск ошибки не завершения рекурсивного вызова функции.

Режим отладки ветвей предполагает, что для выполнения на каждом шаге отладки выбираются операторы независимой ветви графа. Такие операторы всегда выполняются в строгой последовательности, определенной графом программы. Этот режим отладки позволяет четко выделять в параллельной программе линию операторов, которые могут быть выполнены только последовательно один за другим, но при этом не нуждаются в получении каких либо других входных значений из других вершин на данном этапе отладки. При этом цепочка операторов — вершин, выполняемая на отдельном шаге отладки, является логически связанной, все операторы, стоящие в цепи служат одной и той же цели вычисления. Подобная схема отладки не слишком хорошо согласуется с моделью вычислений потоковых параллельных программ, однако здесь не возникает несоответствия между данными, полученными на шагах отладки, и данными, вычисленными при выполнении программы. Режим отладки ветвей выполняет односвязную последовательность команд за один шаг, что в определенных случаях может сократить время проверки программы.

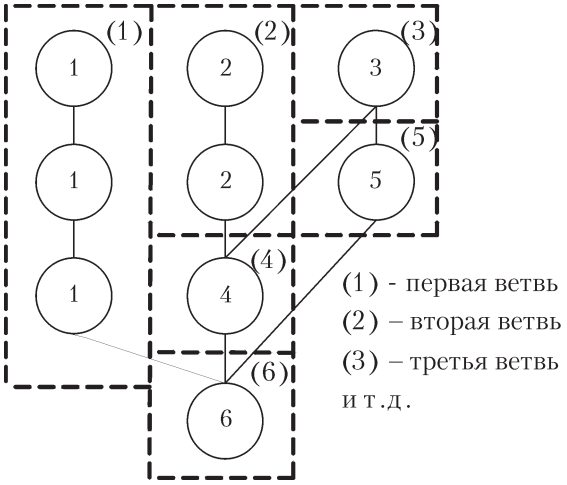


Рис. 2. Пример выделения ветвей в программном графе

Описанные режимы отладки реализованы в среде разработки программ на функционально-потоковом параллельном языке ПИФАГОР. Каждый из режимов обеспечивает использование графического и текстового представления программы

Графическое представление программы при отладке состоит в следующем. Перед началом вычисления любой функции строится ее информационный граф, при выполнении оператора отмечается соответствующая ему вершина графа. Каждая вершина содержит доступную пользователю информацию об операторе и результате его вычисления.

Текстовое представление отладки оперирует списком функций или сверткой функций. Список функций показывает историю шагов отладки, сохраняя текст выполняемой функции на каждом шаге. Свертка функции отображает этот же процесс более компактно, запоминая только предыдущее и вычисленное состояние функции.

Перед отладкой программы, вызывается диалог выбора режима отладки.

Здесь пользователь определяет режим отладки, способ отображения процесса отладки и ряд других опций. «Функция для запуска» — это функция, которая будет запущена для отладки, она выбирается из списка, который формируется автоматически. «Ар-

гумент» — входной параметр для запускаемой функции, он задается пользователем. Функции, не имеющие входного параметра, помечаются в списке функций восклицательными знаками.

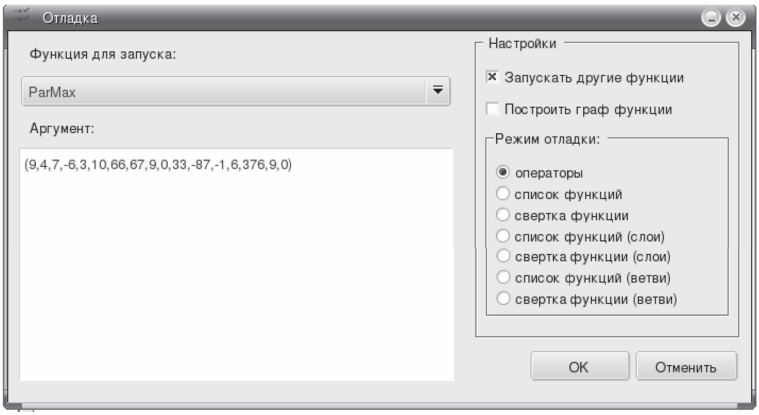


Рис. 3. Диалог выбора режима отладки

Если флаг «Запускать другие функции» отключен, то другие функции не будут запускаться для отладки, это дает возможность сосредоточиться на выполнении выбранной функции, не переключая внимание на просмотр ненужных вычислений. Если флаг включен, отладка проходит полный путь, начиная с запуска, выбранной пользователем функции. Флаг «Построить граф функции» позволяет по желанию пользователя построить граф функции, тогда во время отладки программы будет доступен как граф функции, так и текстовое отображение процесса отладки.

Режимы «операторы», «список функции», «свертка функции» — это различные способы отображения режима пошаговой отладки, «список функции (слои)», «свертка функции (слои)» — различные способы отображения режима отладки слоев. И соответственно «список функции (ветви)», «свертка функции (ветви)» — это различные способы отображения режима отладки ветвей.

Таким образом, режим пошаговой отладки имеет один дополнительный способ текстового отображения — «операторы», по сравнению с режимами отладки ветвей и отладки слоев. Данный способ отображения подобен большинству стандартных способов отображения процесса отладки для последовательных про-

грамм. Существует текстовое поле, содержащее всю программу, по которому перемещается цветовой указатель, показывающий, какой оператор выполняется на данном шаге. Также на главном окне располагается дополнительное текстовое поле, отображающее список всех вычисленных операторов, в виде: 1) исходная строка оператора, 2) строка оператора после подстановки в нее всех значений, вычисленных на предыдущих шагах, 3) вычисленное значение оператора. Программный оператор, вычисляемый на текущем шаге, ставится в конец списка.

В своей пользовательской части такой способ отображения отладки отличается от аналогичных способов отображения отладки последовательных программ путем выполнения операторов. Если для последовательной программы операторы выполняются строго в том порядке, в каком они записаны в тексте программы, то для параллельной программы такая ситуация, скорее является исключением, здесь последовательность вычисления операторов, определяется не расположением в тексте, а с помощью предварительного построения схемы программного графа. Таким образом, оператор, стоящий в n-ой строке, может быть вычислен, ранее оператора, стоящего в $(n-m)$-ой строке.

Подобная ситуация, не противоречит модели вычислений потоковых параллельных программ, но является не лучшей для восприятия процесса отладки пользователем. В связи с этим, были разработаны такие способы текстового представления отладки как создание списка функций и свертка функций, применимые ко всем трем описанным режимам отладки.

Создание списка функций предполагает, что на первом шаге отладки текстовое поле дополняется телом функции, запущенной для отладки, в котором все операторы, вычисляемые на данном шаге, выделяются зеленым цветом. После получения вычисленных значений в поле текста снова дублируется тело функции, но уже с подставленными в него полученными на этом шаге отладки значениями, которые также выделяются цветом. При переходе к следующему шагу отладки поле текста снова дополняется функцией, в которую подставляются вычисленные значения, причем функция содержит и все предыдущие полученные значения. Таким образом, текстовое поле содержит список функций, где каждая последующая функция содержит те операторы, которые не были выполнены на предыдущих шагах,

и также содержит все данные, которые уже были вычислены, выделенные цветом.

Свертка функции отличается от списка функций, только формой отображения информации, получаемой при выполнении шагов отладки. Если была выбрана свертка функции, то на экране отображается исходное тело функции, которое не меняется при отладке, а используется для выделения тех операторов, которые выполняются на данном этапе отладки. Кроме этого, отображается и текущее состояние функции, в котором записаны все предыдущие вычисления, и также добавлены данные, полученные на текущем шаге, последние выделяются цветом. Таким образом, начиная с некоторого шага отладки, исследуемая функция начинает «свертываться». На момент возврата результата вычислений функции, тело функции содержит только записи вида: значение >> идентификатор.

Оба рассмотренных способа текстового отображения процесса отладки имеют как достоинства, так и недостатки. Свертка функции отображает процесс отладки более компактно и наглядно, чем список функций, но при этом не позволяет просмотреть историю всех изменений и переходов.

Графическое отображение отладки выполнено посредством визуализации программного графа отлаживаемой функции. Графическое отображение используется совместно с текстовым отображением, оно может быть включено по желанию пользователя. В программном графе еще не вычисленные вершины имеют серый цвет, вычисленные — зеленый. По нажатию на любую вершину графа появляется информация об исходной строке соответствующего этой вершине оператора, и вычисленном значении оператора, если оно существует. Каждая вершина отображает тип соответственного ей оператора, посредством специального символа. Граф функции всегда располагает операторы-вершины одного слоя на одном уровне, не зависимо от того, какой режим отладки был выбран.

Каждый из рассмотренных режимов отладки и способов отображения процесса отладки реализован практически и может быть использован для отладки функционально-потоковых параллельных программ.

Эффективность каждого отдельного режима зависит от свойств исполняемой программы. Так, простая функция может

содержать только одну ветвь, и это значит, что при выборе последнего режима все вычисления будут выполнены за один шаг, тогда как отладка по слоям, в этом же случае, будет полностью аналогична пооператорной отладке. Плюсы и минусы описанных режимов находятся в зависимости и от восприятия процесса отладки пользователем. Но, в общем случае, пооператорная отладка позволяет сосредоточиться на каждом программном операторе, отладка ветвей выполняет прямые последовательности команд за один шаг, что может сократить время проверки программы, отладка слоев дает представление об уровне параллелизма рассматриваемой функции.

Различные текстовые способы отображения отладки предоставляют такие возможности, как сохранение истории вычислений и вызовов функций, выделение текущих результатов отладки. Графическое представление отладки позволяет увидеть историю вычисления отдельной функции и эффективно проследить все последствия возникновения ошибки, наглядно выделяет ветви и слои, показывает уровень параллелизма функции.

Заключение

Исследование методов отладки и отображения информации при отладке функционально-потоковых параллельных программ позволило выделить ряд основных направлений развития инструментальных средств, повышающих эффективность процесса разработки. Предлагаемые методы реализованы в рамках экспериментальной среды. Полученная реализация повышает эффективность отладки и позволяет более наглядно представлять параллельные вычисления, обеспечивая их различное отображение. Дальнейшие исследования, проводимые на основе разработанной среды, позволят выделить наиболее эффективные методы отладки функционально-потоковых параллельных программ.