УДК 004.4'2

Морылев Р.И. (научный сотрудник, Южный федеральный университет)

Инструмент определения распараллеливаемых циклов

В статье рассматриваются условия возможности параллельного выполнения циклов на машинах с архитектурой SIMD и MIMD, как с общей, так и с распределенной памятью. Представлен автоматический инструмент, реализующий описанный анализ в Диалоговом высокоуровневом оптимизирующем распараллеливателе.

Ключевые слова: Распараллеливание циклов, информационная зависимость, SIMD, MIMD, общая память, распределенная память.

1. Введение

В последние годы многопроцессорные вычислительные системы вытеснили однопроцессорные. Но, несмотря на прогресс в области аппаратного обеспечения, создание программ для таких систем остается сложной задачей, требующей особых знаний и навыков от программистов. В то время, когда преобладали однопроцессорные системы, было написано огромное количество последовательного кода, отказаться от которого по причине сложности и дороговизны переписывания невозможно. Поэтому возникает задача нахождения параллелизма в имеющихся последовательных программах. Известно, что большую часть времени выполнения программы занимают циклы, поэтому в первую очередь необходимо уметь находить параллелизм в циклах.

В рамках проекта Диалогового высокоуровневого оптимизирующего распараллеливателя (ДВОР) реализован инструмент, позволяющий автоматически определять возможность параллельного выполнения для любого цикла в программе. Он может быть использован как при разработке автоматически распараллеливающих компиляторов, так и в консультационных целях при ручном распараллеливании. Отметим, что данный инструмент позволяет проверить лишь корректность распараллеливания, а вопросы его целесообразности и эффективности возлагаются на пользователя.

2. Граф информационных связей

Граф информационных связей (ГИС) – ориентированный граф, вершины которого соответствуют вхождениям переменных, а дуга соединяет пару вершин (u,v) если выполняются условия [1]:

1. эти вхождения обращаются к одной и той же ячейке памяти (т.е. порождают информационную зависимость);
2. на графе потока управления программы существует путь от u к v.

Пример 1.

```
for(i = 1; i < N; ++i)
    A[i] = A[i-1];
```

Вхождения переменной `A` в теле цикла порождают информационную зависимость, поскольку оба обращаются к ячейке памяти `A[1]`: вхождение `A[i]` - на первой итерации цикла, а `A[i-1]` - на второй.

Дуги информационных зависимостей бывают четырех видов, в зависимости от типов начала и конца:

- от генератора к использованию – истинная информационная зависимость;

- от использования к генератору – антизависимость;
- от генератора к генератору – выходная зависимость;
- от использования к использованию – входная зависимость.

Информационная зависимость между вхождениями называется циклически независимой (loop independent dependence), если эти вхождения обращаются к одной и той же ячейке памяти на одной и той же итерации цикла [1]. В противном случае дуга называется циклически порожденной.

3. Условия распараллеливания циклов

Цикл называется распараллеливаемым, если его итерации можно выполнять одновременно. Для определения возможности параллельного (независимого) выполнения итераций цикла на архитектурах SIMD и MIMD используется ГИС.

Доказано, что итерации цикла можно выполнять параллельно, если на ГИС не существует циклически порожденных рассматриваемым циклом дуг зависимости [1, с. 82]. Но следует отметить, что это условие является лишь достаточным, более точный набор условий задается архитектурой и типом памяти целевой системы.

На машинах с общей памятью допускается наличие циклически порожденных дуг входной зависимости (от использования к использованию). На машинах с распределенной памятью возможно параллельное выполнение итераций цикла, даже если на ГИС существуют циклически порожденные дуги анти- или входной зависимости, но при этом необходимо предварительно разослать данные по вычислительным узлам.

Для того, чтобы итерации цикла можно было выполнять параллельно на машине с архитектурой SIMD, необходимо выполнение следующих условий [3]:

1) Границы цикла должны быть известны на этапе компиляции. (1)
2) В теле цикла могут присутствовать только операторы присваивания и циклы DO с единственным входом и единственным выходом. (2)
3) На ГИС рассматриваемого цикла не должно быть дуг зависимости, ведущих от вхождений, выполняющихся раньше, к вхождениям, выполняющимся позже. (3)

На Рисунке 1 – копия экранной формы ДВОР с примером визуализации графа информационных связей.

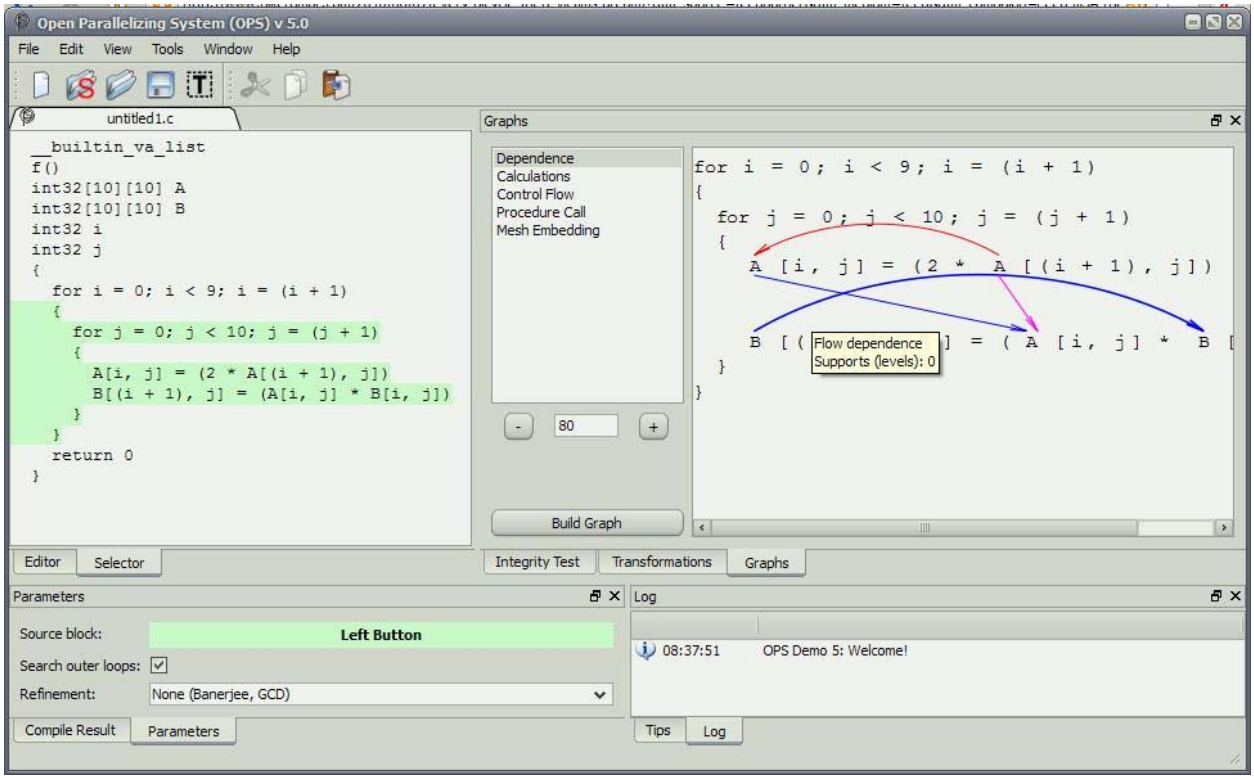

Рис. 1. Пример ГИС в ДВОР. На данном графе дуга потоковой зависимости между операторами является циклически независимой, остальные три дуги – циклически порожденными внешним циклом. Поэтому внутренний цикл распараллеливается, а внешний – нет.

Сложность при распараллеливании представляют циклы, в теле которых присутствуют вызовы функций. В общем случае итерации таких циклов параллельно выполнять нельзя, так как вызываемая функция может иметь побочные эффекты (изменение глобальных переменных, вывод в файл и т.д.). В таких случаях необходимо дополнительно проводить межпроцедурный анализ. Если в результате такого анализа обнаружится, что функция не имеет побочных эффектов, то рассматриваемый цикл можно выполнять параллельно.

4. Реализация инструмента в ДВОР

Внутреннее представление ДВОР [4] является деревом. Каждому циклу исходной программы соответствует узел внутреннего представления. Любой узел дерева внутреннего представления ДВОР можно пометить. Этот механизм используется для пометки параллельных циклов. После применения алгоритма определения параллельности каждый цикл может быть помечен набором меток, в зависимости от архитектуры машины, на которой итерации этого цикла можно выполнять параллельно:

1) SISM — архитектура SIMD с общей памятью.
2) SIDM — архитектура SIMD с распределенной памятью.
3) MISM — архитектура MIMD с общей памятью.
4) MIDM — архитектура MIMD с распределенной памятью.

После прохода алгоритма все циклы разделяются на две группы:

1. Не помеченные – циклы, параллельное выполнение которых невозможно ни на одной из архитектур.
2. Помеченные — циклы, итерации которых можно параллельно выполнять на одной или нескольких архитектурах.

Данные метки могут в дальнейшем использоваться другими компонентами ДВОР: генераторами кода, визуализацией и т.д.

Например, генератор MPI-кода может использовать эту информацию, чтобы определять, для каких циклов в программе можно генерировать параллельный код, и в каких случаях необходимо генерировать код для предварительной рассылки данных. Далее приводится пример визуализации меток в форме отчета.

ГИС в ДВОР строится автоматически. На Рисунке 2 изображена копия экранной формы ДВОР с визуализацией ГИС для одного из фрагментов программы-решателя composeV4_100_corr_phi из пакета ACELAN [2]. Автоматическое построение ГИС позволяет быстро и без участия пользователя определить возможность параллельного выполнения для каждого из циклов в программе.

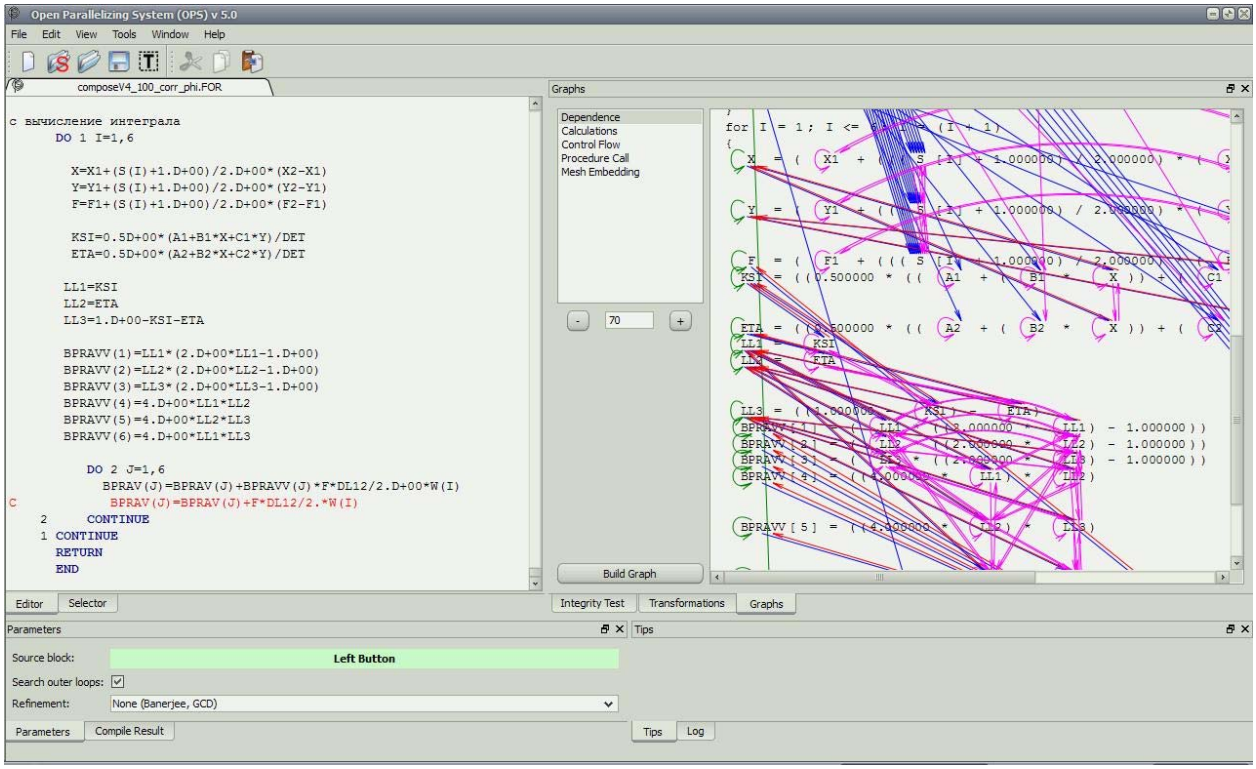

Рис. 2. Визуализация автоматически построенного ГИС для фрагмента программы-решателя composeV4_100_corr_phi из библиотеки ACELAN.

5. Визуализация результатов

Программа, определяющая параллельность циклов, в результате своей работы формирует отчет. В заголовке отчета приводится суммарная информация об общем количестве циклов в программе,

количестве распараллеливаемых и нераспараллеливаемых циклов. После заголовка следует подробный список циклов программы. Циклы нумеруются и группируются по функциям, в которых они встречаются. Для краткости выводятся только заголовки циклов в порядке их следования в программе.

Каждому заголовку цикла предшествует одна из следующих меток: SEQUENTIAL или PARALLEL(<метки архитектур>). Метка SEQUENTIAL говорит о том, что помеченный ею цикл нельзя выполнять параллельно. Метка PARALLEL – что итерации цикла можно выполнять параллельно только на машинах с архитектурой, указанной в скобках (SISM, SIDM, MISM или MIDM). Если цикл нельзя выполнять параллельно на одной или нескольких архитектурах, то после заголовка цикла выводится список причин, по которым это невозможно. При этом возможны следующие причины:

- тело цикла содержит вызовы функций с побочными эффектами;
- тело цикла имеет больше одной точки входа или выхода;
- цикл не является каноническим;
- тело цикла содержит операторы, отличные от операторов присваивания;
- цикл порождает зависимость по данным.

Эта информация может быть использована в дальнейшем для повышения параллельности путем выполнения преобразований над программой.

Если цикл возможно выполнять параллельно при определенных условиях, то после метки архитектуры перечисляются все такие условия. На момент написания данной статьи в программе реализовано автоматическое определение следующих условий:

- необходима предварительная рассылка данных;

- необходима приватизация переменных (указывается список переменных);
- необходима редукция (указывается переменная и операция).

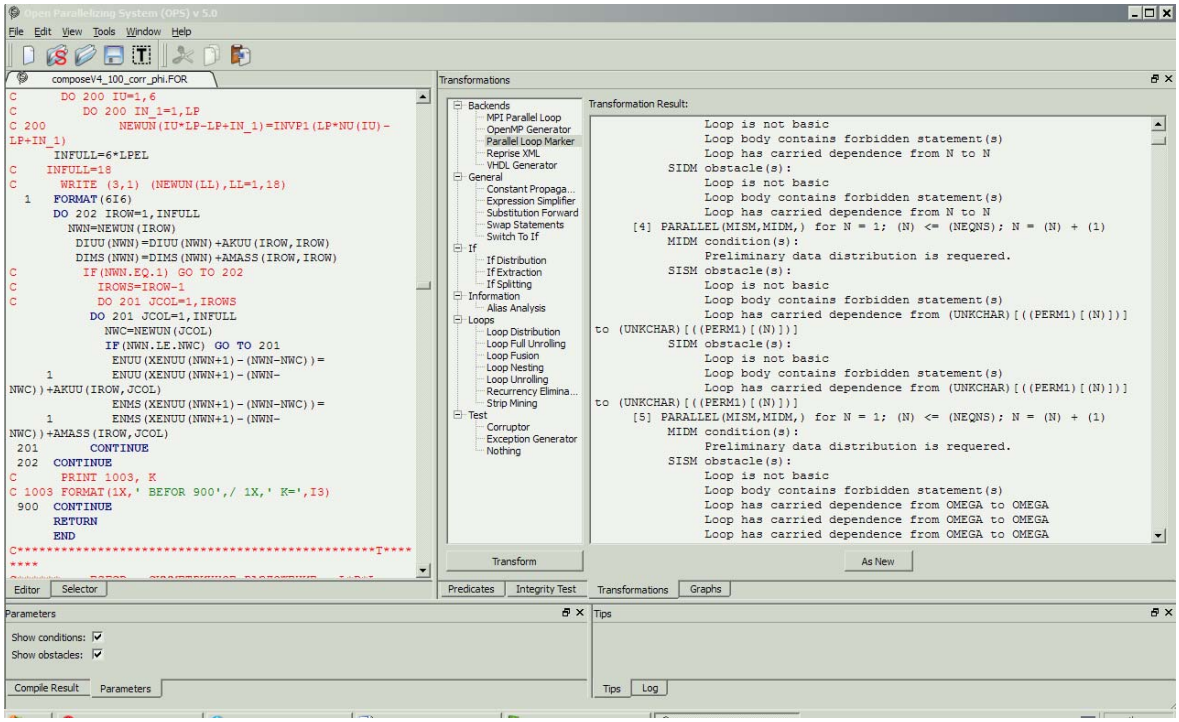

Рис. 3. Экранная форма программы, определяющей распараллеливаемость цикла в ДВОР. Слева - исходная программа, справа – отчет об автоматическом определении распараллеливаемых циклов.

На Рисунке 3 приведена копия экранной формы с фрагментом отчета о распараллеливаемости циклов в программе composeV4_100_corr_phi. На ней видно, что цикл под номером 4 помечен метками MISM и MIDM. Ниже для метки MIDM указано, что для распараллеливания требуется предварительная рассылка данных. Для меток SISM и SIDM выведен список причин, по которым распараллеливание невозможно: цикл не простой; в теле цикла содержатся операторы отличные от оператора присваивания; цикл порождает информационную зависимость.

5. Литература

1. Kennedy K., Allen J. Optimizing Compilers for Modern Architectures: A Dependence-Based Approach. // San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 2001

2. Наседкин А.В., Скалиух А.С., Соловьев А.Н. Пакет ACELAN и конечно-элементное моделирование гидроакустических пьезопреобразователей // Известия ВУЗов. Северо-Кавказский регион. Естеств. науки. 2001. Спецвыпуск. Математическое моделирование. С.122-125.

3. Lamport L. The parallel execution of DO loops// Commun. ACM.- 1974.- v.17, N 2, p. 83-93.

4. Петренко В.В. Внутреннее представление Reprise распараллеливающей системы. // Труды четвертой международной конференции «Параллельные вычисления и задачи управления» PACO’2008. Москва, 27-29 октября 2008г. с. 1241-1245.

Morylev R.I.

Parallel loop determination tool

The paper consider the conditions of parallel execution of do loops on SIMD and MIMD machines both with shared and distributed memory. The description of the automatic tool implementing described analysis in High-level dialogic optimizing parallelizer is presented.

Keywords: loop parallelization, information dependency, SIMD, MIMD, shared memory, distributed memory.