УДК 681.3

ОТКРЫТАЯ РАСПАРАЛЛЕЛИВАЮЩАЯ СИСТЕМА 2006

Б.Я. Штейнберг
Ростовский государственный университет
Россия, 344006, г. Ростов-на-Дону, ул. Б. Садовая, 105
E-mail: steinb@ns.math.rsu.ru

З.Я. Нис
Ростовский государственный университет
Россия, 344006, г. Ростов-на-Дону, ул. Б. Садовая, 105
E-mail: irishrover@mail.ru

В.В. Петренко
Ростовский государственный университет
Россия, 344006, г. Ростов-на-Дону, ул. Б. Садовая, 105
E-mail: petrenko@jeo.ru

Д.Н. Черданцев
Ростовский государственный университет
Россия, 344006, г. Ростов-на-Дону, ул. Б. Садовая, 105
E-mail: denis_tch@mail333.com

Р.Б. Штейнберг
Ростовский государственный университет
Россия, 344006, г. Ростов-на-Дону, ул. Б. Садовая, 105
E-mail: romanofficial@yandex.ru

А.М. Шульженко
Ростовский государственный университет
Россия, 344006, г. Ростов-на-Дону, ул. Б. Садовая, 105
E-mail: sanders@antaresrostov.ru

Ключевые слова: распараллеливающие компиляторы, преобразования программ
Key words: parallelizing compilers, program transformations

В данной статье отражены изменения, произошедшие в Открытой распараллеливающей системе (ОРС) с момента конференции РАСО ‘2001 [1]. Открытая распараллеливающая система предназначена для автоматического распараллеливания программ с процедурных языков программирования на параллельные компьютеры. С помощью ОРС могут быть разработаны: распараллеливающие компиляторы и конверторы, диалоговые распараллеливающие системы, компиляторы с параллельных языков программирования, системы анализа последовательных программ с точки зрения возможностей распараллеливания, программы для обучения параллельному программированию, и другие компилирующие продукты.

OPEN PARALLELIZING SYSTEM 2006 / B.J. Steinberg (Rostov State University, 105, B. Sadovaja st., Rostov-on-Don 344006, Russia, E-mail: steinb@ns.math.rsu.ru), Z.J. Nis (Rostov State University, 105, B. Sadovaja st., Rostov-on-Don 344006, Russia, E-mail: steinb@ns.math.rsu.ru), V.V. Petrenko (Rostov State University, 105, B. Sadovaja st., Rostov-on-Don 344006, Russia, E-mail: steinb@ns.math.rsu.ru), D.N.

Cherdantsev (Rostov State University, 105, B. Sadovaja st., Rostov-on-Don 344006, Russia, E-mail: steinb@ns.math.rsu.ru), R.B. Steinberg (Rostov State University, 105, B. Sadovaja st., Rostov-on-Don 344006, Russia, E-mail: steinb@ns.math.rsu.ru), A.M. Shulzhenko (Rostov State University, 105, B. Sadovaja st., Rostov-on-Don 344006, Russia, E-mail: steinb@ns.math.rsu.ru). The paper describes modifications in the Open Parallelizing System (OPS) after the PACO '2001. The Open Parallelizing System has intended for automatic parallelization procedure languages programs. Many compiler products may be created with the OPS: parallelizing compilers and converters, dialog parallelizing systems, systems of analysis of serial programs, e-learning material about parallel programming and others.

1. Введение

Предназначение. Открытая распараллеливающая система (ОРС) предназначена для автоматического распараллеливания программ с процедурных языков программирования на параллельные компьютеры (на данный момент на входе ОРС программы на языке Си). С помощью ОРС могут быть разработаны: распараллеливающие компиляторы, распараллеливающие конверторы (в языки высокого уровня), диалоговые распараллеливающие системы, компиляторы с параллельных языков программирования, программы для анализа последовательных программ с точки зрения возможностей распараллеливания, программы для обучения параллельному программированию, и другие компилирующие продукты. Предполагается, что преобразования из библиотеки ОРС можно будет комбинировать для различных архитектур: конвейерных, векторных, SIMD, MIMD, VLIW, DataFlaw, мультитредовых, программируемых …

Основные части ОРС – внутреннее представление, графовые модели программ, библиотека оптимизирующих/распараллеливающих преобразований программ, дополнительные функции компиляции, интерфейс с развитой системой визуализации.

Информация об ОРС частично представлена в Интернете www.ops.rsu.ru.

2. Внутреннее представление ОРС

Основой Открытой распараллеливающей системы является внутреннее представление. Внутреннее представление ориентируется на многоязыковый вход (фортран, паскаль, си), на удобство разработки преобразований программ и на возможность генерации кода различных целевых архитектур. С момента работы [1] о ОРС внутреннее представление изменилось и вся система была переписана. На сегодняшний день есть парсер и препроцессор языка Си в это внутреннее представление. Внутреннее представление допускает возможность разработки в дальнейшем парсеров с языков Паскаль и Фортран. Это внутреннее представление может быть трансформировано для работы и с другими процедурными, в том числе и объектно-ориентированными, языками.

Внутреннее представление (ВП) – это структура данных для хранения информации о программе, а также алгоритмы, облегчающие выполнение преобразований программ [16]. Новое ВП ОРС разработано в соответствии со следующими требованиями. Внутреннее представление должно быть:

- универсальным, т.е. позволять хранить программы с нескольких процедурных языков, рассматриваются Си (C99), Фортран (Fortran 77) и Паскаль (ISO 10206:1990);
- базисом для построения, анализа информационных зависимостей и написания преобразований программ, при этом похожие свойства процедурных языков унифицировать, а специфичные сохранять;
- расширяемым для подключения компиляторов переднего плана с новых языков программирования к ОРС;
- удобным для генерации машинного кода, кода с вызовом MPI, OpenMP.

Внутреннее представление спроектировано с применением Design Patterns [23] и реализовано на языке C++. Были проведены работы по интеграции внутреннего представления и нового компилятора переднего плана ОРС с языка Си. Разработан механизм передачи информации между различными частями ОРС в виде специальных пометок в узлах внутреннего представления. Это позволяет единообразно организовать передачу контекстно-зависимой информации между графовыми моделями программ, преобразованиями и генераторами выходного кода. Реализованы отладочные средства и средства проверки целостности структуры внутреннего представления во время работы программы.

Новое внутреннее представление Открытой распараллеливающей системы представляет связную древовидную структуру и является композицией из четырех деревьев: типов, идентификаторов, выражений и операторов. За основу для конструирования деревьев внутреннего представления взят паттерн “Composite” [23]. Каждое из четырех деревьев ВП хранит информацию о какой-то части программы. Дерево типов содержит информацию о типах в языке программирования, в том числе о пользовательских типах. Дерево идентификаторов – об именах в программе, их распределение по областям видимости. Дерево выражений представляет выражения в программе, узлы – операции и операнды. Дерево операторов представляет операторы в программе.

Прежде чем переходить к рассмотрению деревьев, следует упомянуть о способе конструирования узлов. Для этой цели применяется паттерн “Factory” [23]. Поскольку язык C++ не содержит сборщик мусора, программист вынужден самостоятельно следить за освобождением памяти. При проектировании ВП было принято два важных решения:

- Все объекты-узлы всех деревьев создаются в динамической памяти.
- Узел при удалении ответственен за освобождение памяти всех своих поддеревьев.

Первое решение обосновано единообразием выделения памяти. И соответственно, исключив из рассмотрения выделение памяти на стеке, позволяет использовать более простую стратегию освобождения памяти. Второе решение было принято после ряда экспериментов. На ранних этапах разработки нового ВП рассматривалась идея организовать ссылки между узлами через «умные указатели» (Smart Pointers). Однако при практическом применении оказалось, что в данной работе использование таких указателей неэффективно.

Первоначальное внутреннее представление ОРС было разработано несколько лет назад [16]. Был написан специальный генератор компиляторов, разбирающий программы с сильно сокращенных подмножеств языков Фортрана, Паскаля и Си. Это позволило начать реализовывать преобразования. На старом внутреннем представлении было реализовано девять распараллеливающих преобразований [17], в том числе два – «Введение временных массивов» и «Растягивание скаляров» автором. Появившийся в результате выполнения этой работы

опыт, позволил обнаружить недостатки старого ВП, и прийти к пониманию каким должно быть новое.

Внутреннее представление, предназначенное для хранения информации о программах на языках высокого уровня должно содержать высокоуровневые конструкции, такие как циклы, определение абстрактных типов данных, модулей.

Большинство конструкций во внутреннем представлении перешли из языка Си. Например, цикл FOR Си и Паскаля (цикл DO Фортрана) во внутреннем представлении называется StmtFor и характеризуется узлом с четырьмя потомками: выражение инициализации (InitExpr), выражение условия (CondExpr), выражение инкремента (IncrExpr) и тела цикла (Body). Таким образом, например, цикл For Паскаля

```
for i := 1 to 10 do
  ...
```

может быть записан во внутреннем представлении:

```
InitExpr: i = 1
CondExpr: i <= 10
IncrExpr: ++i
Body: ...
```

В то же время определение массивов в языке Паскаль является более общим, чем в Си:

```
M : array[1..10, -5..5] of integer;
```

Поэтому узел дерева типов, хранящий массив во внутреннем представлении содержит информацию о верхней и нижней границе массива. Более того, в языке Си понятие многомерного массива вводится индуктивно, так, например, трехмерный массив представляется как массив двухмерных массивов, а двухмерный массив как массив массивов. Во внутреннем представлении ОРС многомерный массив – это один узел в дереве типов, которых хранит информацию о количестве измерений массива и границы индексов каждого.

В новое внутреннее представление, как функции классов, перешли сервисные функции прежнего внутреннего представления:

1) вставить группу операторов
2) удалить группу операторов
3) заменить группу операторов
4) вставить подвыражение в выражение
5) удалить подвыражение
6) заменить подвыражение
7) сгенерировать новое имя переменной
8) сгенерировать новую метку
9) сгенерировать описание нового массива

Внутреннее представление проектировалось так, чтобы при преобразованиях программ сохранялась синтаксическая и семантическая целостность. Так

операторы циклов и ветвления представлены в полном виде. Это означает, что при добавлении, например, оператора IF сразу создается выражение (пустое) для условия, а также пустые блоки для THEN и ELSE веток оператора. В частности, это решает проблему «кочующего ELSE». Таким образом, невозможна ситуация, когда оператор IF (в терминах языка Паскаль) имеет только ELSE, без THEN. Или когда в блоке имеется два ELSE, но только один IF.

В процессе работы над новым внутренним представлением были рассмотрены внутренние представления наиболее известных распараллеливающих систем: Polaris [25] и SUIF [26].

Система Polaris, разработанная в центре суперкомпьютерных исследований университета Иллинойс, является распараллеливающим и оптимизирующим конвертером фортран-фортран. Внутреннее представление этой системы, как и OPC, реализовано на языке C++.

Прежде всего, следует отметить, что внутреннее представление Polaris ориентировано только на один язык программирования – Фортран 77, это во многом определило его архитектуру. Например, отсутствует такая важная для современных процедурных языков (Си, Паскаль, Java) конструкция, как блок операторов (список операторов с ассоциированной с ним областью видимости имен). Пометки в узлах ВП не предусмотрены. Вместо них разработчики предоставили стек временных объектов, ассоциированный со всей программой (единицей трансляции), поэтому разработчик программы не может ссылаться на определенный элемент программы (оператор, операцию, вхождение) неким стандартным образом.

Граф информационных зависимостей в системе Polaris неразрывно связан с внутренним представлением. Это означает, что при выполнении преобразований приходится его постоянно перестраивать, чтобы поддерживать его информацию о зависимостях в актуальном состоянии. В OPC граф информационных зависимостей строится до выполнения преобразований заново. Это особенно актуально, потому что OPC на момент написания этой работы уже содержит пять графовых моделей: граф информационных связей, решетчатый граф, граф вычислений, граф вызова подпрограмм и управляющий граф. Синхронное перестроение всех этих графов заняло бы слишком много времени и усложнило бы реализацию преобразований внутреннего представления.

SUIF (Stanford University Intermediate Format) – программная система для исследования распараллеливающих и оптимизирующих преобразований [26]. Эта система также реализована на C++.

Внутреннее представление SUIF рассчитано на языки Си и Фортран 77. Однако поддержка языка Фортран изначально в систему не закладывалась, программа с него конвертером f2c переводится в Си, а только потом во внутренний формат. Очевидно, что при таком подходе специфичная для программ на Фортране информация никак не отражается во внутреннем представлении.

В качестве основного способа расширения функциональности в системе SUIF используется написание нового прохода (compiler pass). При этом, в системе не предусмотрено никаких способов использовать имеющиеся, а также добавлять новые, алгоритмы поиска и анализа информационных зависимостей в программе.

Внутреннее представление OPC имеет ряд схожих черт с внутренним форматом SUIF, например, основанный на паттерне “Composite” [23] способ представления программ и использование пометок узлов для передачи информации между отдельными частями компилятора.

3. Интерфейс и система визуализации

Интерфейс и система визуализации позволяют в диалоге производить цепочки преобразований, являются основой электронного учебника, позволяют анализировать возможности распараллеливания программ, автоматически выводя на экран графовые модели программ.

На следующей экранной форме видно, что программист выбрал преобразование “Iteration Space Splitting”, открыл свою программу (панель слева), отметил гнездо циклов FOR, а также два вхождения переменной “a”. Затем нажал кнопку “Transform”, и в окне справа получил преобразованную программу:

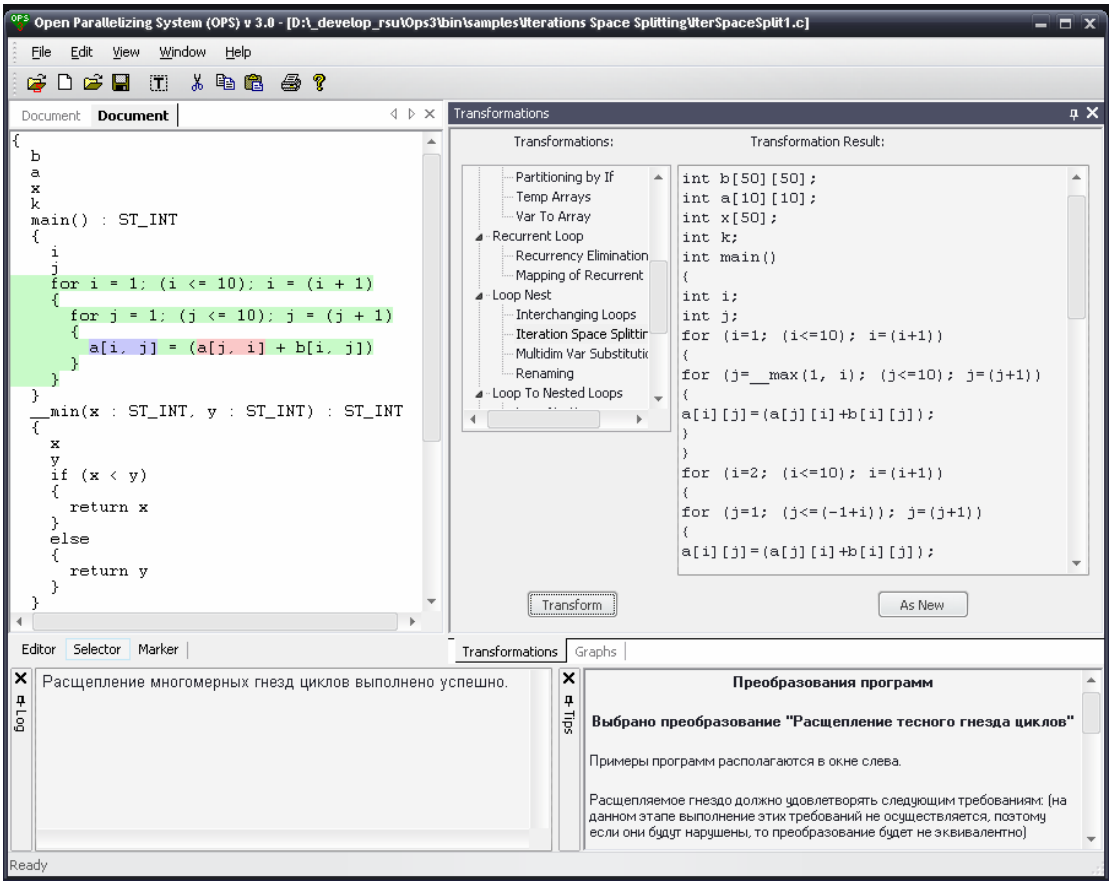

Рис. 1. Интерфейс OPC. Диалоговое выполнение преобразований.

Далее, после нажатия “As New”, результат преобразования переместится с правой панели на левую, что позволит продолжить применение преобразований. В правом нижнем окне после выбора преобразования отображается его краткое описание и инструкция по применению. Нажатие кнопки «Справка» открывает в отдельном окне раздел электронного учебника, в котором имеется полное описание выбранного преобразования.

4. Управление семантической корректностью преобразований

Так как большой класс преобразований оперирует фактически исходными текстами программ, то для таких преобразований становится актуальной проблема сохранения статической семантической корректности – преобразованная программа должна оставаться семантически корректной.

Анализ показал, что даже в простых преобразованиях возможны неочевидные проблемы с семантикой. Рассмотрим преобразование «раскрутка цикла», заключающееся в замене цикла со счетчиком N-кратным повторением тела цикла, где N – число итераций в исходном цикле. Если тело цикла содержит операторы досрочного перехода к следующей итерации (оператор continue в языке Си), то после применения преобразования эти операторы окажутся вне цикла, что является нарушением семантических ограничений в Си. В случае, когда преобразуемый цикл вложен в другой цикл, семантика нарушена не будет, но логика программы изменится. Таким образом, разработчик данного преобразования должен учитывать особые случаи, возникающие, когда в теле цикла встречаются «опасные» для преобразования операторы и конструкции. Для преобразований, являющихся композициями других преобразований, «особых случаев» будет еще больше, а их обнаружение будет представлять проблему.

В Открытой распараллеливающей системе ведутся работы по автоматизации контроля семантической корректности преобразований программ. Система контроля корректности включает в себя разработку внутреннего представления на основе AST (Abstract Syntax Tree) для эффективного представления программ на Си, разработку языка описания преобразований, разработку описания статической семантики процедурных языков.

5. Графовые модели программ

В ОРС реализованы следующие графовые модели программ:

- Граф информационных связей.
- Граф вычислений (автоматическая конвейеризация цикла с синхронизацией).
- Решетчатый граф программы.
- Управляющий граф программы.
- Граф вызовов подпрограмм.

В ОРС преобразования программ используют как традиционный граф информационных связей (программных зависимостей), так и решетчатый граф программ (который активно исследуется в работах P. Feautrier, В.В. Воеводина, Н.А. Лиходеда и группы ОРС). При этом, граф информационных зависимостей для фрагмента программы строится как на основе неравенств Банержи и НОД теста, так и на основе либо элементарных, либо простых снизу решетчатых графов. Построение графа информационных зависимостей на основе решетчатых графов оказывается более точным. Решетчатый граф хранится в памяти с помощью покрывающих функций.

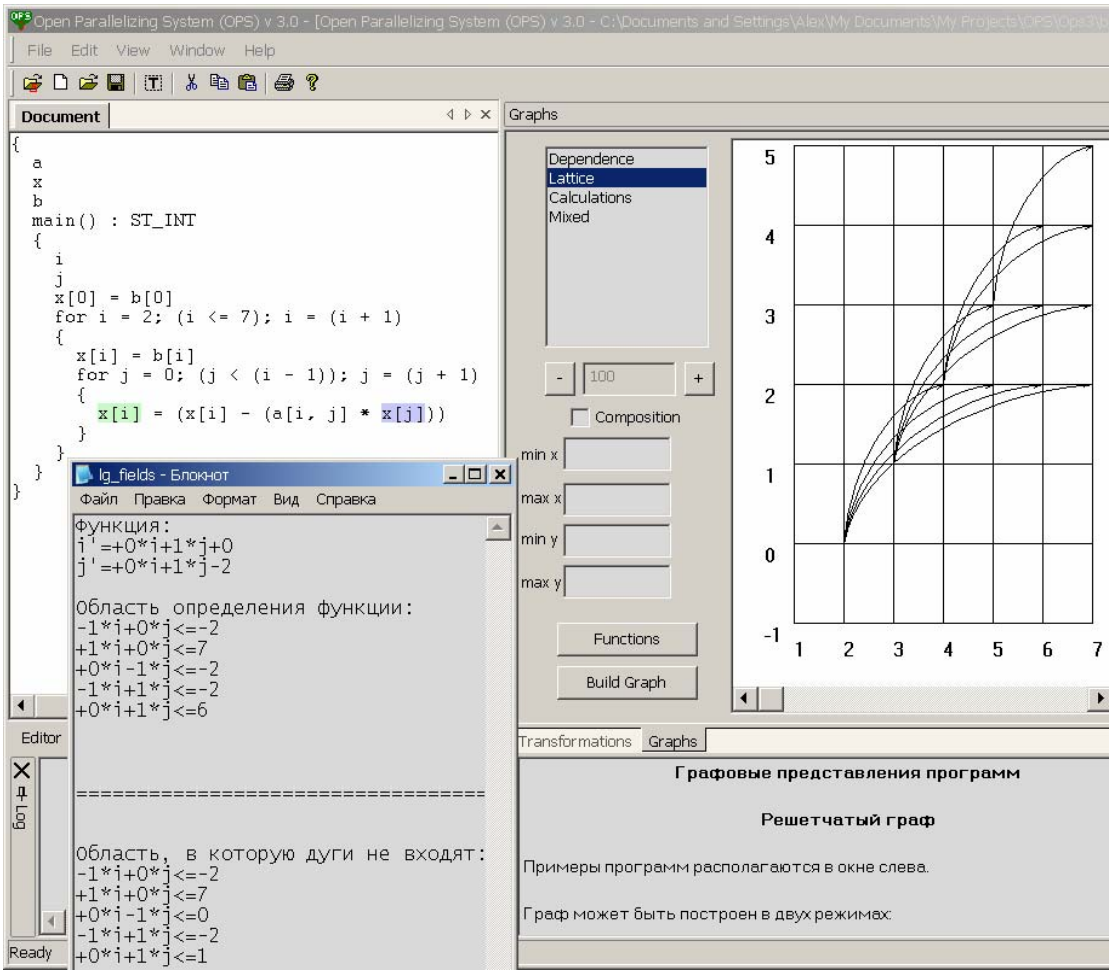

Рис. 2. Построение и визуализация решетчатого графа в OPC. В левом верхнем окне выделены два вхождения переменных, для которых строится граф. Справа вверху визуализация соответствующего графа. Слева внизу функции, описывающие этот граф.

6. Библиотека преобразований OPC

Библиотека преобразований программ – это основная ценность OPC. Библиотека преобразований программ на сегодняшний день состоит из более двух десятков преобразований. Это высокоуровневые преобразования, выполняемые во внутреннем представлении. Для сравнения отметим, что в распараллеливающей системе PARAWISE – 7 преобразований, в системе SUIF – 12 преобразований, в системе POLARIS много теоретических наработок, но по сайту сложно судить о степени реализации. Эти преобразования делятся на группы, в зависимости от вида фрагментов исходного текста, к которым они применяются. Написанные преобразования автоматически проверяют условия эквивалентности, основанные на графе информационных связей или на решетчатом графе. В OPC реализованы смешанные вычисления и стандартизация выражений. Это позволяет отождествлять выражения вида k+1+1 и 2+k, что бывает необходимо при определении информационных зависимостей и поиске коэффициентов перед переменными.

Среди описанных преобразований иногда можно выбрать пары таких, которые являются взаимно обратными. Примером таких преобразований являются «подстановка вперед» и «вынос общих подвыражений».

6.1. Подстановка вперед

Это преобразование подставляет правую часть оператора присваивания вместо последующих вхождений левой части этого же оператора (если это не нарушает равносильность) [22]. В частном случае, когда правая часть оператора присваивания является константой, это преобразование называется протягиванием констант. В результате такого преобразования исчезают некоторые дуги информационной зависимости, что может способствовать распараллеливанию, и уменьшается объем выполняемого кода.

Пример. Данный фрагмент программы из двух операторов присваивания

```
10   X(K) = A(I+1)
20   B = X(K)
```

равносилен следующему

```
10   X(K) = A(I+1)
20   B = A(I+1)
```

Теорему об условиях эквивалентности подстановки вперед можно найти в [7].

В ОРС реализована подстановка вперед без проверки эквивалентности. Так же, в ОРС реализована подстановка вперед для переменных и констант интервального типа.

Интервальная подстановка вперед подставляет не только правую часть оператора присваивания вместо последующих вхождений левой части этого же оператора (если это не нарушает равносильности), а так же подставляет и значение для погрешности.

Пример. Данный фрагмент программы

```
c13_err = 0;
c13 = 10;
…
y_err = z_err[1+k]+ c13_err;
y = z[1+k] + c13;
```

равносилен следующему

```
c13_err=0;
c13=10;
…
y_err=z_err[(1+k)]+0;
y=z[(1+k)]+10;
```

6.2. Вынесение общих подвыражений

Это преобразование заключается в поиске одинаковых выражений (подвыражений) и замене их выполнения обращением к специально заведенной переменной.

Пример. Данный оператор

```
d=(a+b+c)*(a+b);
```

будет заменен на фрагмент

```
x=a+b;
d=(x+c)*x;
```

Таким образом, можно сократить число выполняемых операций.

В ОРС выражения выносятся с учетом ассоциативности и коммутативности.

6.3. Преобразования циклов

Преобразованиям циклов в ОРС уделялось наибольшее внимание. В первую очередь реализована канонизация циклов, т.е. приведение циклов к виду, в котором счетчик цикла равен 1, а левая граница равна либо 0, либо 1 (бывает потребность как в одном, так и в другом случае). Канонизация цикла всегда является эквивалентным преобразованием и носит вспомогательный характер для возможного применения других преобразований.

Разбиение циклов требует проверки специальных условий эквивалентности. Иногда для возможности применения разбиения циклов выполняются вспомогательные преобразования «расщепление вершин» («введение дополнительных массивов»), «растягивание скаляров». («Замена скалярной переменной массивом»), «перестановка операторов».

Слияние циклов может рассматриваться, как преобразование, обратное к перестановке циклов.

Вынос инварианта из цикла является одним из наиболее эффективных преобразований последовательной оптимизации.

Перестановка циклов в тесном гнезде может улучшить программу с точки зрения обращения к памяти и с точки зрения распараллеливания (в т.ч. и конвейеризации, и векторизации).

Есть преобразования одномерных циклов к двумерным: «гнездование цикла» и «разрыв итераций». Эти преобразования могут создавать в программе циклы определенной длины. Это может быть нужно, например, для того, чтобы количество итераций распараллеливаемого цикла совпадало с количеством процессоров.

6.4. Преобразования, основанные на решетчатых графах

Все описанные выше преобразования либо эквивалентны всегда, либо их эквивалентность проверяется с помощью графа информационных связей. В Открытой распараллеливающей системе реализованы некоторые преобразования программ, основанные на решетчатых графах, которые не могут быть основаны на традиционном графе информационных связей [8], [9]. К таким преобразованиям относятся:

1) расщепление многомерных гнезд циклов в виде последовательности тесных гнезд;
2) подстановка индексных переменных, на основе расщепления в виде последовательности тесных гнезд циклов;
3) экспансия массивов, на основе расщепления в виде последовательности тесных гнезд циклов.

Кроме того, на основе решетчатых графов определяются циклы, итерации которых могут выполняться независимо (циклы ParDo по решетчатым графам [10]).

7. Оценки точности вычислений

В рамках ОРС реализован режим компиляции с автоматической оценкой погрешности начальных данных и округления. Получен конвертер, преобразующий пользовательскую программу на языке Си в новую программу, которая помимо запрограммированных пользователем вычислений выдает и оценки погрешностей проведенных вычислений.

Для расчета погрешности производится замена входной переменной парой чисел (Например, переменная A заменяется на пару (A, Aerr), где A – приближение к истинному значению, а Aerr – оценка относительной погрешности) и переопределением арифметических операций [21].

Пример. Пусть есть фрагмент программы

```
Double a, b, c;
…..
 a = a*b + c;
```

Пусть вычисления проводятся в десятичных числах с 4 значащими цифрами. Тогда после применения рассматриваемого преобразования имеем:

```
double a, b, c;
double aerr, berr, cerr;
……
aerr=(aerr*1 + berr*1+0.0005)*(a*b)/(a*b+c)+cerr*(c/(a*b+c))+0.0005;
 a = a*b +c;
```

При построении формул оценок для выражений, содержащих вызовы функций, иногда требуется больше усилий. Например, пусть надо построить по-

грешность для вызова функции double f(double d, int i). Тогда в программу будет добавлена функция double ferr(double d, double derr, int i, double r), в теле которой будут переопределены операции. Результат работы функция ferr – оценка ошибки функции f.

Так же отметим, что при таком преобразовании программы меняется ввод данных. Например, если в программе была одна переменная A, и вводилось ее значение, то теперь должно вводится еще и значение для начальной погрешности.

8. Частичные вычисления

Это преобразование позволяет на этапе компиляции упрощать, в том числе и уменьшать время выполнения функций, если известны значения некоторых входных параметров.

В OPC реализован моновариантный метод частичных вычислений [27, 20].

Пример. Пусть есть функция

```
int func(int n, int m)
{    int l;
   l= 5*n+7*m+10;
  return  l
};
```

И есть вызов этой функции

```
  .. = .. func(10, x) ..
```

Тогда после применения данного преобразования в программе появится еще одна функция

```
int func_1(int m)
{    int l;
   l= 7*m+60;
  return  l
};
```

А исходный вызов будет заменен вызовом новой функции

```
  .. = .. func_1(x) ..
```

9. Конвейерные и многоконвейерные вычисления

На данный момент в OPC реализован расчет таких параметров конвейеризации циклов как задержки и интервал инициализации итераций. По одномер-

ному циклу определяется наличие зависимостей, препятствующих конвейеризации (нерегулярная истинная зависимость, выходная зависимость и антизависимость). В случае их отсутствия, строится регулярный граф алгоритма (РГА), определяется критический путь на этом графе и обратные дуги, порожденные регулярными истинными зависимостями. ОРС позволяет проанализировать все эти характеристики визуально, с помощью модуля визуализации РГА. Также ОРС позволяет сгенерировать таблицу расписания по описанным выше характеристикам цикла и его РГА, а затем визуализировать эту таблицу. Модуль расчета синхронизации конвейера ориентирован на архитектуру перестраиваемого конвейера [24], но может быть адаптирован и к любой другой конвейерной архитектуре (см. рис. 3).

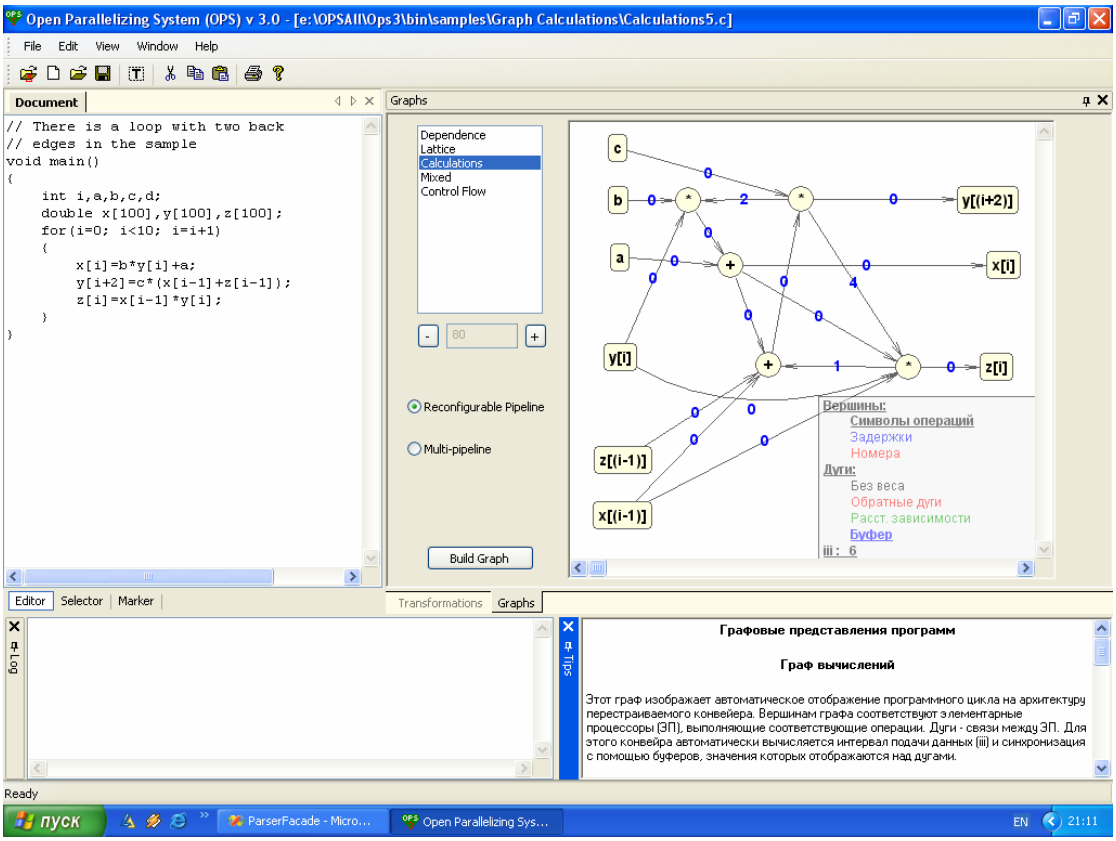

Рис. 3. Визуализация редуцированного графа алгоритма и автоматического расчета параметров конвейера. Каждой операции из цикла в левой части экрана соответствует вершина в РГА (в правой части экрана), а каждая дуга указывает на передачу информации между соответствующими операциями. Для каждой вершины указана операция, которую она выполняет, а для каждой дуги указана величина задержки (буфера), которая необходима для синхронизации передачи информации между соответствующими операциями.

Группой ОРС ведутся исследования в области многоконвейерных вычислительных систем. Исследуемая в работах модель многоконвейерной вычислительной системы допускает выполнение нескольких программных фрагментов на нескольких конвейерах одновременно. Например, такими фрагментами могут быть одномерные конвейеризуемые циклы. Тесное двумерное гнездо циклов может быть рассмотрено как набор таких одномерных циклов: каждой итерации внешнего цикла ставиться в соответствие некоторый конвейер. На данный момент получены формулы вычисления задержек между моментами начала рабо-

ты конвейеров [12]. Описаны некоторые высокоуровневые преобразования, приводящие фрагмент программы к виду, удобному для отображения на многоконвейерную архитектуру [13]. Программный модуль расчета задержек между стартами конвейеров находится в процессе разработки и в ближайшее время будет подключен к ОРС. Этот модуль позволит по тесному двумерному гнезду циклов определять возможность отображения каждой итерации внешнего цикла на параллельно функционирующие конвейеры. В случае наличия зависимостей между итерациями внешнего цикла, будет рассчитана оптимальная задержка между стартами конвейеров, которая позволит синхронизировать пересылки данных между конвейерами.

10. Заключение

В Открытой распараллеливающей системе ведутся работы над распараллеливанием рекуррентных циклов, над оптимизацией кода, содержащего условные операторы, над динамическими преобразованиями программ. Ведется работа с индуктивными переменными. Разрабатывается преобразование «инверсия цикла». Предполагается разработка архитектурно-ориентированных комбинаций преобразований. Ведутся исследования в теории автоматического распараллеливании программ [1-19], [28].

На сегодняшний день в Открытой распараллеливающей системе из внутреннего представления генерируется код на Си, т.е. ОРС работает, как конвертор. ОРС позволяет в диалоге преобразовывать код языка Си и привести его к виду, удобному для распараллеливания. Высокий уровень преобразований и внутреннего представления позволяют оптимизировать и генерировать код на различные целевые архитектуры. В ОРС автоматически определяются циклы ParDO, т.е. циклы, итерации которых могут выполняться независимо [10]. Автоматически определяются конвейеризуемые циклы, автоматически рассчитывается синхронизация конвейерных вычислений (интервал инициализации итераций, значения задержек). Выполнена экспериментальная автоматическая генерация Си с процедурами MPI. Сделан конвертор программ Си к виду, автоматически вычисляющему оценки погрешности вычислений.

Список литературы

1. Штейнберг Б.Я. Открытая распараллеливающая система // Труды Международной конференции «Параллельные вычисления и задачи управления» PACO ‘2001. Москва, 2-4 октября 2001 г. М.: Институт проблем управления им. В.А. Трапезникова РАН, 2001. С. 214-220.
2. Штейнберг Б.Я. Распараллеливание рекуррентных циклов с условными операторами // АиТ. 1995, № 6. С. 176-184.
3. Штейнберг Б.Я. Оптимальные параллельные переразмещения двумерных массивов // Программирование. 1993. № 6. С. 81-88.
4. Штейнберг Б.Я. Бесконфликтные размещения массивов при параллельных вычислениях // Кибернетика и системный анализ. 1999. № 1. С. 166-178.
5. Штейнберг Б.Я. Подстановка и переименование индексных переменных в многомерных циклах // Известия вузов. Северокавказский регион. Юбилейный выпуск. 2002. С. 94-99.
6. Штейнберг Б.Я. Распараллеливание рекуррентных программных циклов // Информационные технологии. 2004. № 4. С. 16-23.

7. Штейнберг Б.Я. Математические методы распараллеливания рекуррентных программных циклов на суперкомпьютеры с параллельной памятью. Ростов-на-Дону: Издательство Ростовского университета, 2004. 192 с.
8. Шульженко А.М. Расщепление многомерных циклов для эффективного распараллеливания // Труды Всероссийской научно-технической конференции «Параллельные вычисления в задачах математической физики». Ростов-на-Дону, 21-25 июня 2004 г. С. 186-194.
9. Шульженко А.М. Решетчатый граф и использующие его преобразования программ в Открытой распараллеливающей системе // Научный сервис в сети Интернет: технологии распределенных вычислений. Труды Всероссийской научной конференции. Новороссийск, 19-24 сентября 2005. М.: Изд-во МГУ, 2005. С. 82-85.
10. Шульженко А.М. Автоматическое определение циклов ParDo в программе // Известия вузов. Северокавказский регион. Естественные науки. Приложение 11'05. С. 77-88.
11. Шульженко А. М. Преимущества определения информационных зависимостей в программе с помощью решетчатых графов // XIII Международная конференция «Математика. Экономика. Образование». Тезисы докладов. Ростов-на-Дону, 2005. С. 137 ISBN 5-94153-097-8
12. Штейнберг Р.Б. Вычисление задержки в стартах конвейеров для суперкомпьютеров со структурно процедурной организацией вычислений // Искусственный интеллект. Научно-теоретический журнал. Институт проблем искусственного интеллекта НАНУ. Украина, Донецк: ДонДИШИ, «Наука и Освита». 2003. № 4. С. 105-112.
13. Штейнберг Р.Б. Некоторые оптимизирующие преобразования программ для компьютеров, допускающих многоконвейерную обработку данных // Научный сервис в сети Интернет: технологии распределенных вычислений. Труды Всероссийской научной конференции. Новороссийск, 19-24 сентября 2005 г. М.: Изд-во МГУ, 2005. С. 85-87.
14. Петренко В.В. Внутреннее представление ОРС3 // Научный сервис в сети Интернет: технологии распределенных вычислений. Труды Всероссийской научной конференции. Новороссийск, 19-24 сентября 2005 г. М.: Изд-во МГУ, 2005. С. 106-108.
15. Черданцев Д.В. Автоматическая оценка погрешности // Научный сервис в сети Интернет: технологии распределенных вычислений. Труды Всероссийской научной конференции. Новороссийск, 19-24 сентября 2005 г. М.: Изд-во МГУ, 2005. С. 81-82.
16. Штейнберг Б.Я., Макошенко Д.В., Черданцев Д.Н., Шульженко А.М. Внутреннее представление в Открытой распараллеливающей системе. // Искусственный интеллект. Научно-теоретический журнал. Институт проблем искусственного интеллекта НАНУ. Украина, Донецк: ДонДИШИ, «Наука и Освита». 2003. № 4. С. 89-97.
17. Штейнберг Б.Я., Черданцев Д.Н., Науменко С.А., Бутов А.Э., Петренко В.В. Преобразования программ для открытой распараллеливающей системы // Искусственный интеллект. Научно-теоретический журнал. Институт проблем искусственного интеллекта НАНУ. Украина, Донецк: ДонДИШИ, «Наука и Освита». 2003. № 3. С. 97-104.
18. Штейнберг Б.Я., Бутов А.Э., Науменко С.А., Петренко В.В., Черданцев Д.Н., Штейнберг Р.Б., Шульженко А.М. Полуавтоматическое распараллеливание на основе ОРС // Труды Всероссийской научно-технической конференции «Параллельные вычисления в задачах математической физики». Ростов-на-Дону, 21-25 июня 2004 г. С. 186-194.
19. Штейнберг Б.Я., Арутюнян О.Э., Бутов А.Э., Гуфан К.Ю., Морылев Р., Науменко С.А., Петренко В.В., Тузаев А., Черданцев Д.Н., Шилов М.В., Штейнберг Р.Б., Шульженко А.М. Обучающая распараллеливанию программа на основе ОРС // Научно-методическая конференция «Современные информационные технологии в образовании: Южный федеральный округ». Ростов-на-Дону, 12-15 мая 2004 г. С. 248-250.
20. Ершов А.П. Смешанные вычисления: потенциальные применения и проблемы исследования. http://ershov.iis.nsk.su/archive/eaindex.asp?lang=1&did=2596
21. Мак-Кракен Д., Дорн У. Численные методы и программирование на Фортране / Пер. с англ. М.: Мир, 1977 584 с.
22. Векторизация программ // Векторизация программ: теория, методы, реализация / Сборник переводов статей. М.: Мир, 1991. С. 246-267.
23. Гамма Э., Хэлм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб: Питер, 2001. 368 с.
24. Каляев А.В. Программирование виртуальных параллельных проблемно-ориентированных суперкомпьютеров в структуре универсальных суперкомпьютеров с массовым параллелизмом // Международная научно-техническая конференция «Интеллектуальные многопроцессорные системы». Таганрог, 1-5 сентября 1999 г. Сборник трудов. С. 27-39.
25. Faigin K., Hoeflinger J., Padua D., Petersen P., Weatherford S. The Polaris Internal Representation.

26. Aigner G., Diwan A., Heine D., Lam M., Moorey D., Murphy B., Sapuntzakis C. The SUIF Program Representation.
27. Jones N.D., Gomard C.K., Sestoft P. Partial evaluation and automatic program generation. Prentice-Hall, Inc., 1993.
28. www.ops.rsu.ru