мережі у вигляді лінійного кола постійного струму без втрат і у вигляді нелінійного кола постійного струму з втратами забезпечують адекватне представлення передавальної мережі ОЕС України в математичних моделях мережі лібералізованого ринку електроенергетики України

1. *Борисенко А.В., Саух С.Є.* Рівноважна модель вводу генеруючих потужностей в умовах недосконалої конкуренції // Новини енергетики. – 2009. – №11. – С. 36—39; №12. – С. 23—39.
2. *Костюковский Б.А., Шульженко С.В., Гольденберг І.Я., Власов С.В.* Методи та засоби дослідження перспектив розвитку електроенергетики в умовах впровадження ринкових відносин // Проблеми загальної ененергетики. —2002.— № 2. – С. 6—13.
3. *Hobbs B.F., Metzler C.B., Pang J.S.* Strategic gaming analysis for electric power system: an MPEC approach // IEEE Transactions on Power Systems. – 2000. – Vol. 15, № 2. – P. 638—645.
4. *Саух С.Є., Борисенко А.В., Джигун О.М.* Модель сети магистральных линий электропередачи в задачах планирования развития электроэнергетических систем.– Электронное моделирование. – 2014. – 34, № 4. – С. 3-14.
5. *Саух С.Е.* Модель конкурентного равновесия на рынке электроэнергии с улучшенной адекватностью математического описания генерирующих компаний, системного оператора и электрической сети.– Электронное моделирование. – 2016. – 38, № 4. – С. 49-6414.
6. *Handbook of* Mathematical Functions with Formulas, Graphs and Mathematical Tables / M.Abramowitz, I.Stegun. – National Bureau of Standards, USA. – 1972. – 1046p.

Поступила 10.04.2017р.

УДК004.94

А. А.Чемерис, г. Киев

ОСОБЕННОСТИ ТЕКСТА КОМПЬЮТЕРНЫХ ПРОГРАММ В ПРОЦЕССЕ РАСПАРАЛЛЕЛИВАНИЯ

Abstract. The paper considers some questions of preliminary preparation of C source for automatic parallelization and optimization. It is shown that some parallelism cannot be found due to not appropriate text syntax of the source of the program. Some ways of text correction are shown.

Введение

Автоматическое распараллеливание программ, представленных на языке высокого уровня, есть один из эффективных путей повышение производительности компьютерных систем. Однако, не любая программа,

обладающая параллелизмом может быть распараллелена. Это объясняется тем, что не любая языковая конструкция может быть распознана компилятором на предмет выявления параллелизма.

Одним из известных подходов при статическом распараллеливании программ есть такой подход, при котором исходный текст программы на языке высокого уровня трансформируется также в исходный текст, содержащий параллельные языковые конструкции, и который далее может быть откомпилирован стандартными компиляторами соответствующей вычислительной системы[1]. Такой подход называется трансформация source-to-source (преобразование исходный текст – исходный текст программы) [2]. Примером такого подхода для трансформации программ служит пакет PIPS, который представляет собой инструмент для компиляции Си и Фортран программ. Результатом работы этого пакета есть распараллеленный исходный код программы [3]. Положительные стороны такого подхода состоят в следующем:

- методы распараллеливания, как правило, являются преобразованиями исходного кода программ;
- результат всех шагов оптимизации и компиляции, может быть выраженоператорамиСи;
- позволяет сравнивать исходные и преобразованные коды, а также, простую трассировку и отладку полученных программ;
- простой инструментарий оптимизации, который является комбинацией трансформаций.

Известно, что различные методы распараллеливания обладают конкретными ограничениями, как методического характера, так и программной реализации. Так, например, методы аффинных преобразований, направленные на распараллеливание циклов с обработкой массивов данных, индексы которых представлены линейными выражениями с постоянными коэффициентами, не работают с индексными выражениями $k*i-d$.

Таким образом, к тексту программ выдвигаются определенные требования, которые требуется соблюсти, иначе данный фрагмент текста не сможет быть распараллелен. Далее рассмотрим несколько примеров языковых конструкций на языке высокого уровня Си.

Работа выполняется в рамках ведомственной темы НАН Украины «Розвитокметодівзниженняенергоспоживанняобчислювальних систем за рахунокоптимізаціїобробкимасивівданих» (2014-2018).

Директивы препроцессора.

Особенностью языка Си являются директивы препроцессора, которые представляют собой инструкции, выполняемые до процесса компиляции программ. Они позволяют изменить текст программы, в зависимости от некоторых условий, вставить текст из другого файла, запретить трансляцию части текста, заменить некоторые текстовые лексемы в программе.

Ниже приведен простейший пример, в котором директивой #define определена лексема COEF.

```
#define  N   100
#define  COEF  2
…….
void main() {
inti;
float a[N], b[N];
…….
for(i = 1; i<N; i++)
            a[i] = b[COEF*i+1]+a[i-1];
……
}
```

В данном фрагменте кода константа COEF без прохода препроцессора может быть воспринята как символьное имя, обозначающее переменную. Соответственно, трансформации цикла, основанные на аффинных преобразованиях, применены не будут.

Особое значение имеют конструкции языка, обеспечивающие мультиплатформенность программ. Это реализовано с помощью директив препроцессора. С их помощью выбираются фрагменты программы, которые затем компилируют для конкретной платформы. Естественно, что распараллеливание исходной программы без обработки препроцессором не имеет смысла.

Таким образом, при применении автоматического распараллеливания, программа на языке высокого уровня Си должна предварительно пройти обработку препроцессором, а только затем проводится распараллеливание.

Указатели

Особенностью программ на языке Си есть использование указателей, в частности для определения динамических массивов. Данный подход широко используется, особенно при обработки очень больших массивов. Соответственно, программы на Си изобилуют специфическими синтаксическими конструкциями, которые затрудняют использование алгоритмов распараллеливания. Применяемые программы при распараллеливании таких исходных текстов не определяют возможного параллелизма из-за синтаксиса обрабатываемых исходных тексов Си программ.

Следующий пример показывает обращение к массиву, когда элемент двумерного массива определяется через указатель на начало массива плюс смещение.

```
int N = 10;  // размерность массива
float B[N][N], *A[];
……
B[i][j] = A[N*i+d];
……
```

Здесь также воспринимается индексное выражение как нелинейное и, соответственно, не будут применены методы аффинных преобразований.

Возможен более сложный вариант записи обращения к массиву:

```
B[i][j] = *(A+N*i+d);
```

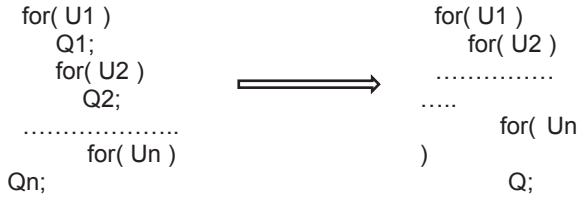
В этом примереиспользуется косвенная адресация. Для правильного восприятия оператор компилятором необходимо преобразовать текст в `B[i][j] = A[N*i+d]` .

Еще один пример использования адресации при обращении к массиву состоит в сложной косвенной адресации вида `A[B[i]]`. Анализируя данный текст, компилятор может потерять зависимость определения элементов массива `B` и их использования в роли индекса массива `A`. Для распараллеливающего компилятора более понятна следующая запись

```
…
BB = B[i];
C = A[BB];
…
```

Представление циклов в программах

Большинство программ распараллеливания ориентированы на обработку циклов программ, представленных в тесногнездовой форме.



Выполнение преобразования в автоматическом режиме затруднено, поскольку требуется не только формальное преобразование текста, а изменение семантики оператора. Для этого требуется рассматривать каждый конкретный случай в отдельности. Затем применять различные модификации преобразования текста. Это может быть объединение циклов с дальнейшим представлением в тесногнездовом виде, расширение границ переменных или количества их измерений, введение новых переменных и т.д.

В качестве такого преобразования можно привести следующий пример. В этом примере оператор вычисления элемента массива A переносится в самый внутренний цикл. Недостатком такого преобразования есть появление условных операторов if, которые проверяют выполнение соответствующего условия. Таким образом, целесообразность применения преобразования к совершенно вложенному циклу (тесногнездовому) следует анализировать в каждом конкретном случае.

```
for ( k = 1; k == N; k++ )
A(k,k) = sqrt (A(k,k)) ;
for( i = k+1; i == N; i++ )
      A(i,k) = A(i,k) / A(k,k);
// перенести во внутренний цикл
for( j = k+1; j == i; j++ )
A(i,j) -= A(i,k) * A(j,k);
```

```
for( k = 1; k == N; k++ )
   A(k,k) = sqrt (A(k,k));
      for( i = k+1; i == N; i++ )
         for( j = k; j == i; j++ )
            if (j==k) A(i,k) = A(i,k) /
A(k,k);
 if (j!=k) A(i,j) -= A(i,k) * A(j,k);
```

Основные особенности такого преобразования операторов:

1. выполнение цикла до/после итерации цикла, в котором выполняется только перенесенный оператор;

2. требуется вставка условных операторов, как фрагментов, обеспечивающих правильное выполнение операторов в новом цикле.

Практически эффективно использовать для несовершенство-вложенных циклов, в котором каждый цикл имеет только один другой цикл, вложенный непосредственно в него. С точки зрения оптимизации программ, эффективно использовать при задаче увеличения локальности вычислений.

Заключение

Статические методы выявления параллелизма работают только с текстом программы и из-за этого имеют ряд ограничений, препятствующих обнаружению параллелизма в программе. Тем не менее, этот тип методов анализа широко применяется в существующих системах автоматизации распараллеливания. Для его эффективного применения требуется соответствие текста распараллеливаемой программы возможностям методов обнаружения параллелизма. Есть возможность автоматического преобразования программ. Особенно это интересует разработчиков source-to-source систем распараллеливания.Но, как показано, не все языковые конструкции могут быть автоматически обработаны. С другой стороны, возможно создание системы правил, обязывающих программиста писать код программы так, а не иначе. Этот путь на практике не имеет перспективы. Практически, результатом, как правило, есть рождение новых версий (диалектов) известных языков высокого уровня, в которые добавлены некоторые директивы, указывающие на наличие параллелизма в том или ином фрагменте программы.

Таким образом, нужен дополнительный анализ и преобразование текста программ, а именно, 1). выполнение прохода перед анализом на распараллеливание препроцессора; 2). на синтаксис программ при их написании наложить определенные требования; 3). проводить преобразования циклов while и do-while к циклам for; 4). возможно проводить дополнительный анализ указателей в программе.

1. *Chunhua Liao, Daniel J. Quinlan, Jeremiah J. Willcock and Thomas Panas*, Semantic-Aware Automatic Parallelization of Modern Applications Using High-Level Abstractions, Journal of Parallel Programming, International Journal of Parallel Programming, October

2010, Volume 38, Issue 5, pp 361–378.
2. *UdayBondhugula, A. Hartono, J. Ramanujan, P. Sadayappan.* A Practical Automatic Polyhedral Parallelizer and Locality Optimizer. // ACM SIGPLAN Programming Languages Design and Implementation (PLDI), Jun 2008, Tucson, Arizona.
3. *Amini, M., et al.*: PIPS Is not (just) Polyhedral Software. In: International Workshop on Polyhedral Compilation Techniques (IMPACT'11). Chamonix, France. – April, 2011.

Поступила 3.04.2017р.

УДК 519.711

Ю.Г. Куцан, Г.А. Иванов, Киев

ЭВОЛЮЦИЯ РЫНКА ЭЛЕКТРИЧЕСКОЙ ЭНЕРГИИ УКРАИНЫ. ПУТИ ТРАНСФОРМАЦИИ И ОЖИДАЕМЫЕ РЕЗУЛЬТАТЫ

Abstract. The paper describes the stages of transformation of relations between participants of the Ukrainian electric energy market after the declaration of its independence in 1991. The description of the current electric energy market in Ukraine and the stages of its formation and liberalization are given.

После распада Советского Союза в 1991 Украине досталась в наследство модель рынка электрической энергии с вертикально-интегрированными государственными компаниями, которые на региональном уровне вырабатывали, передавали, распределяли и поставляли электрическую энергию потребителям в регионах. Как такового оптового рынка электрической энергии не существовало, и полностью отсутствовала конкуренция среди участников рынка.

С 1996 года и по сегодняшний день в Украине функционирует оптовый рынок на основе модели «Элетроэнергетического Пула Англии и Уэльса», но с некоторыми особенностями. Эта модель основана на принципах обязательной продажи всей выработанной и импортированной электрической энергии единому покупателю, а покупка электрической энергии для дальнейшей поставки или экспорта осуществляется только у этого единого покупателя. Оператором оптового рынка выступает государственная компания «Энергорынок», деятельность которой полностью зарегулирована. Рынок организован на основе договора между всеми членами рынка. На оптовом рынке конкурируют только производители электрической энергии, вырабатываемой на угольных энергоблоках. Цена на их электроэнергию складывается по результатам конкурентного отбора оператором оптового рынка. Производители электрической энергии на атомных, гидравлических, теплоэлектроцентралях и из возобновляемых источников энергии, продают электрическую энергию в оптовый рынок по установленным государством