УДК: 004.43

ИНСТРУМЕНТАЛЬНАЯ ПОДДЕРЖКА ТРАНСЛЯЦИИ И ВЫПОЛНЕНИЯ ФУНКЦИОНАЛЬНО-ПОТОКОВЫХ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

И.В. Матковский, А.И. Легалов

Рассматриваются инструментальные средства, обеспечивающие трансляцию и выполнение функционально-потоковых параллельных программ, написанных на языке программирования Пифагор. Представлена общая структура среды разработки и выполнения, принципы функционирования ее отдельных компонент, форматы промежуточных представлений.

Ключевые слова: функционально-потоковое параллельное программирование, трансляторы, языки программирования.

Введение

Потребность в повышении производительности при решении сложных прикладных задач привела к появлению разнообразных архитектур параллельных вычислительных систем (ПВС), каждая из которых имеет свои особенности, проявляющиеся в подходах к управлению вычислениями [1]. Это ведет к разнообразию методов параллельного программирования и затрудняет разработку переносимых приложений. В ряде случаев перевод программы с одной ПВС на другую либо требует серьезной переработки программы, либо приводит к падению эффективности вычислений.

Одним из возможных путей разработки переносимых приложений является использование методов архитектурно-независимого параллельного программирования, базирующихся на концепции неограниченного параллелизма. Предполагается, что программа разрабатывается для ПВС с бесконечными ресурсами, что позволяет при ее написании не акцентировать внимание на ресурсных ограничениях и связанных с ними механизмах синхронизации. Основной упор при разработке таких программ делается на логику функционирования при сохранении максимального параллелизма, а привязка к реальным вычислительным ресурсам конкретной ПВС осуществляется после верификации и отладки.

Для использования данного подхода предложена функционально-потоковая парадигма параллельного программирования, на основе которой разработан язык программирования Пифагор [2]. Написанная на нем программа задает вычисления в виде информационного графа, управление вычислениями в котором осуществляется по готовности данных. Для организации вычислений были разработаны соответствующие инструментальные средства:

- транслятор исходных текстов функций в промежуточное представление, определяемое как реверсивный информационный граф (РИГ) [3];
- генератор управляющего графа (УГ), формирующий по РИГ данные, необходимые для выполнения вычислений в событийном процессоре [3,4];
- компоновщик программы, собирающий воедино множество функций, необходимых для выполнения требуемой функции;
- интерпретатор, предназначенный для выполнения функционально-потоковых параллельных программ.

Обобщенная структура системы инструментальной поддержки представлена на рисунке 1.

Транслятор

Транслятор (trans2) переводит исходный текст заданной функции в РИГ. Во входном файле при этом может находиться несколько функций. На полученном РИГ каждой функцией можно проводить различные оптимизационные преобразования, такие как избавление от дублирующихся констант, удаление избыточных операторов и другие.

Файл, обрабатываемый транслятором, состоит из объектов двух типов: функций и именованных констант. Каждый объект обрабатывается отдельно от других и сохраняется в отдельном файле промежуточного представления в каталоге модульной библиотеки. Синтаксическая ошибка в любом месте файла с программой полностью прерывает работу транслятора. При этом результаты для уже оттранслированных констант и функций не сохраняются.

Каждой функции соответствует свой РИГ, в котором, как минимум, имеются две вершины: вершина для входного аргумента и вершина для возвращаемого значения.

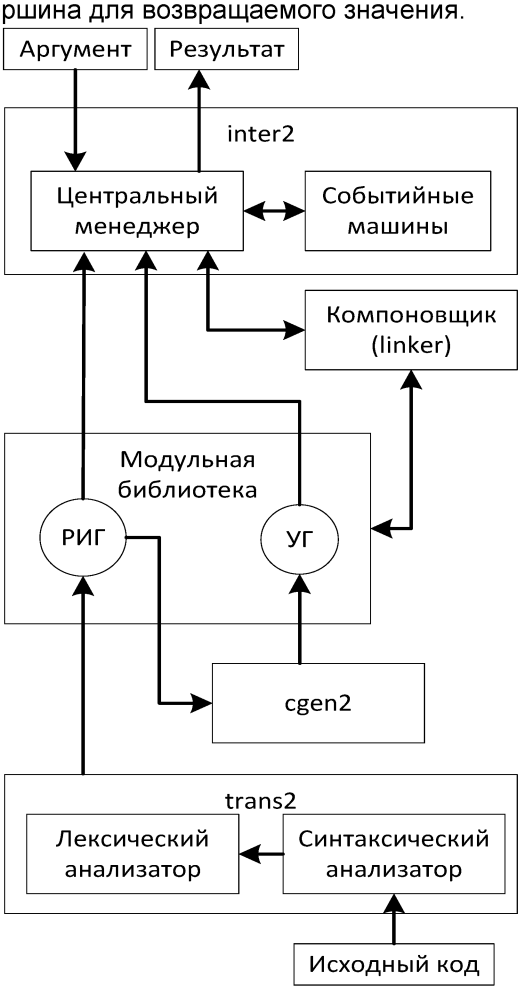

Рисунок 1 – Структура системы инструментальной поддержки

Генератор управляющего графа

Данное средство (обозначено на рисунке 1 как cgen2) осуществляет анализ реверсивного информационного графа, формируя на его основе управляющий граф. Необходимость УГ обуславливается спецификой организации вычислений. Выполняющий их событийный процессор осуществляет обход УГ, реагируя на сигналы, поступающие от его узлов. При этом РИГ используется только для выборки и хранения данных. Подход позволяет оптимизировать управление вычислениями, меняя только УГ при неизменном РИГ

Компоновщик

Предназначен для объединения воедино функций, используемых при выполнении вычислений заданной функции (linker на рисунке 1). Исходными данным для компоновщика является запускаемая функция, на основании анализа которой ищутся функции, вызываемые из нее. Данный процесс рекурсивно продолжается для всех найденных функций. В результате формируется список функций, совместно используемых для выполнения вычислений. Каждая функция встречается в сформированном списке только один раз, несмотря на возможное повторное ее обнаружение в иерархии вызовов.

Аналогичным образом компоновщик собирает и константы, которые в ходе трансляции также отделяются и хранятся как отдельные программные объекты.

Интерпретатор

Интерпретатор осуществляет выполнение заданной функции. Используя список функций и констант, полученных в результате компоновки, он загружает для каждой функции ее РИГ и УГ в оперативную память, образуя тем самым законченную функционально-потоковую программу. После этого начинается процесс выполнения (интерпретации) программы. Список загруженных компоновщиком РИГ и УГ хранит центральный менеджер. Выполнение каждой функции осуществляется своим событийным процессором [3, 4].

Процесс интерпретации начинается с создания первого событийного процессора и ему передаются ссылки на РИГ и УГ начальной функции. Управление вычислений начинается с получения из УГ списка сигналов, задающих начальные события, после чего начинается передача этих сигналов в вершины УГ. Изменение состояния этих вершин ведет, в конце концов, к переходу в состояние, инициирующее запуск вычислений, определяемых соответствующими вершинами РИГ. Процесс передачи управляющих сигналов по УГ продолжается до тех пор, пока не будет обработана вершина, соответствующая в РИГ оператору “return”. В этом случае функция считается выполненной.

При вызове другой функции формируется новый событийный процессор, который связывается со своими РИГ и УГ. Пример взаимодействия центрального менеджера и двух событийных машин приведен на рисунке 2. Показано, что, помимо загруженного списка РИГ и УГ, центральный менеджер должен хранить список существующих событийных процессоров и поступившие от процессоров сигналы. Для каждого из процессоров, помимо уникального идентификатора и ссылки на сам процессор, нужно хранить указатели на области памяти для аргумента и результата выполнения функции. Указатели будут ссы-

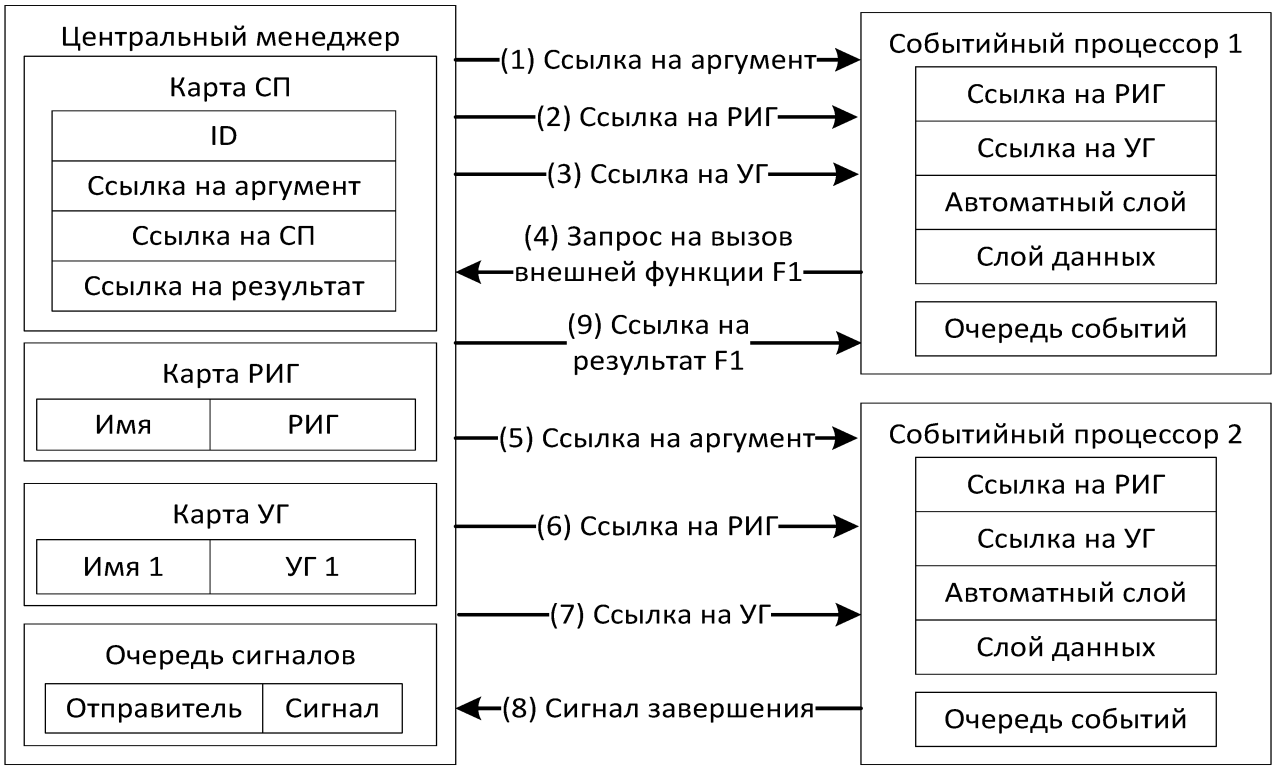

Рисунок 2 – Взаимодействие элементов интерпретатора

латься на области в слоях данных соответствующих событийных процессоров и, помимо прочего, гарантировать сохранность этих областей при удалении самих процессоров. Удаление аргумента можно производить после того, как соответствующий событийный процессор полностью закончил свою работу. Удаление результата можно производить после того, как закончит свою работу процессор, инициировавший порождение данного сигнала.

Реверсивный информационный граф РИГ подразделяется на три части:

- список внешних объектов – перечень имен объектов, использующихся в данной функции и при этом определенных за её пределами;
- список локальных констант – перечень константных значений, определенных в рамках данной функции;
- список вершин – перечень вершин РИГ, соответствующих операторам программы.

Внешние объекты – это функции и константы, использующиеся в данной функции. Как минимум один внешний объект в соответствующем списке РИГ содержится всегда. Это имя самой функции.

Локальные константы задают объекты, размещенные в данной функции. В их список включаются все константные значения встроенных типов (целочисленных, вещественных, символьных) и ярлыки задержанных списков.

В списке вершин содержатся вершины РИГ; каждая из них соответствует одному программно-формирующему оператору языка [2]. Для каждой вершины известны:

- уникальный номер (идентификатор);
- номер задержанного списка, к которому вершина принадлежит;
- тип вершины;
- Узлы РИГ, в которые ведут связи из данной вершины;
- положение задающего вершину оператора.

Рассмотрим пример РИГ для функции получения абсолютного значения числа abs.

```
abs << funcdef X {
        val<< ({(0,X):-}, X);
        key<<((X, 0):[<,>=]):?;
        return<<val: key:.;
}
```

Сохраняемое в файле текстовое представление РИГ будет выглядеть следующим образом:

```
External
0     abs
Local
0     0
1     {1}2
```

```
2     0
id    delay operation  links      positions
0     0     arg                   pos 7 21 8 1
1     1     (---)      loc:0 0    pos 9 8 9 13
2     1     :          1 -        pos 9 13 9 14
3     0     (---)      loc:1 0    pos 9 6 9 19
4     0     (---)      0 loc:2    pos 10 9 10 14
5     0     [---]      < >=       pos 10 15 10 21
6     0     :          4 5        pos 10 14 10 15
7     0     (---)      6          pos 10 8 10 22
8     0     :          7 ?        pos 10 22 10 23
9     0     :          3 8        pos 11 12 11 13
10    0     :          9 .        pos 11 18 11 19
11    0     return     10         pos 10 25 11 7
```

Управляющий граф

Управляющий граф содержит:

- список вершин графа.
- список стартовых сигналов – перечень сигналов, существующих на момент запуска программы и инициализирующих работу с УГ.
- Список динамических связей – перечень связей, формирующихся непосредственно в процессе выполнения программы.

В УГ вершины не различаются между собой по типу; управляющих граф состоит из однотипных элементов, каждый из которых связан с соответствующим автоматом. Если ранее УГ использовался для контроля за передачей управляющих сигналов и их порождением, то сейчас по УГ сигналы лишь передаются; порождаются они в связанных с вершинами УГ автоматах.

Для каждой вершины известны:

- ее уникальный номер;
- номер задержанного списка, в который вложена вершина;
- тип связанного с вершиной автомата;
- номер состояния, в котором данный автомат находится изначально;
- номер вершины РИГ, связанной с соответствующей вершиной УГ;
- список входов, на которые поступают сигналы из данной вершины в формате:

НомерВершины “,” НомерВхода.

В списке стартовых сигналов хранятся сигналы, которые существуют до начала работы интерпретатора. Существование данных сигналов обуславливается структурой функции. Как правило в список стартовых входят сигналы, порождаемые локальными константами функции.

В списке динамических связей содержатся описания связей, которые могут переопределяться по мере интерпретации программы.

УГ функции нахождения абсолютной величины числа выглядит следующим образом:

```
abs
id    delay automat      inodelinks
0     0     arg,0    0   links:1,2;3,2;4,1;
1     1     (---),0  1   links:2,1;
2     1     :,0      2
3     0     (---),0  3   links:9,1;
4     0     (---),0  4   links:6,1;
5     0     [---],0  5   links:6,2;
6     0     :,0      6   links:7,1;
7     0     (---),0  7   links:8,1;
8     0     :,0      8   links:9,2;
9     0     :,0      9   links:10,1;
10    0     :,0      10  links:11,1;
11    0     return,0     11
Signals: (Number, Node, Input)
      0     1     1
      1     2     2
      2     3     1
      3     4     2
      4     5     1
      5     5     2
      6     8     2
      7     10    2
Dynamic links: (Number, Delay list, Node)
      0     1     2
```

Заключение

Разработанные инструментальные средства обеспечивают создание и выполнение функционально-потоковых параллельных программ, что позволяет начать практические эксперименты с применением принципа архитектурно-независимого параллельного программирования.

СПИСОК ЛИТЕРАТУРЫ

1. Легалов, А.И. Об управлении вычислениями в параллельных системах и языках программирования / А.И. Легалов // Научный вестник НГТУ, № 3 (18), 2004. С. 63-72.
2. Легалов, А.И. Функциональный язык для создания архитектурно-независимых параллельных программ / А.И. Легалов // Вычислительные технологии, № 1 (10) - Новосибирск, 2005. - с. 71-89.
3. Легалов, А.И. Событийная модель вычислений, поддерживающая выполнение функционально-потоковых параллельных программ / А.И. Легалов, Г.В. Савченко, В.С. Васильев // Системы. Методы. Технологии. № 1 (13). - 2012. - С. 113-119.
4. Редькин, А.В. Событийное управление выполнением функционально-потоковых параллельных программ / А.В. Редькин, А.И. Легалов // Научный вестник НГТУ, № 3 (32). – 2008. – С. 111-120.

*Ст. преп. **И.В. Матковский**, д.т.н., проф., зав. каф. А.И Легалов, Сибирский федеральный университет, кафедра вычислительной техники.*