7. *Kalman R., Busy R.* New results in linear filtering and prediction theory. //Transaction ASME Journal of Basic Engineering. Series D, v.83, p.95–108.
8. *Harvey A., Koopman S., Shepard N.* State space and unobserved component models. Theory and applications. Cambridge University Press, 2004.

К ВОПРОСУ ОБ АВТОМАТИЗАЦИИ РАСПАРАЛЛЕЛИВАНИЯ ПРОГРАММ

В. Д. Львович, А. Н. Перемотова

ВВЕДЕНИЕ

На сегодняшний день практически все высокопроизводительные системы используют преимущества такой технологии, как параллелизм. И это неслучайно, ведь данная технология позволяет наиболее эффективно использовать мощности современных вычислительных систем. Существование большого количества отлаженного и активно используемого последовательного кода «заставляет мечтать» о его автоматической трансформации в параллельный код. Возможность автоматического распараллеливания сулит очевидные выгоды:

- сокращение времени разработки параллельных программ;
- удешевление процесса разработки параллельных программ;
- снижение требований к квалификации программистов;
- повышение надежности разрабатываемых программ;
- упрощение переноса многих разработанных программ на параллельные компьютеры;
- эффективное проектирование программно-аппаратных комплексов с учетом новых методов распараллеливания.

Цель работы – помочь пользователю в разработке параллельного приложения. В результате исследования были разработаны программные инструменты, позволяющие упростить разработку многопоточных программ. Инструменты позволяют автоматически получить параллельную программу одним из двух способов:

- из последовательной программы посредством расстановки директив OpenMP;
- задать параметры задачи поиска и получить параллельные реализации в нескольких вариантах.

ИНСТРУМЕНТ ANTLR

В работе был использован известный и популярный инструмент ANTLR, который представляет собой генератор анализаторов. ANTLR позволяет определять правила, по которым лексический анализатор

должен токенизировать поток символов, и правила, по которым синтаксический анализатор должен интерпретировать поток токенов. После этого ANTLR может сгенерировать сканер и/или парсер, которые могут быть использованы для решения требуемой задачи.

Беря на себя основную работу по анализу текстов, ANTLR помогает решать различные задачи по преобразованию текстов. В частности, в данной работе потребовалось выполнить следующие преобразования программ:

- вставка в текст С-программы директив OpenMP, при условии удовлетворения текста программы заданным свойствам;
- перевод текста программы, написанного на авторском языке программирования на язык программирования С.

ОТ С-ПРОГРАММЫ К OPENMP-ПРОГРАММЕ

Технология OpenMP является отраслевым стандартом и очень успешным языком параллельного программирования для компьютеров с общей памятью. Значительная часть функциональности OpenMP реализуется при помощи директив компилятору. Они должны быть явно вставлены пользователем, что позволит выполнять программу в параллельном режиме.

OpenMP поддерживает такой стиль программирования, при котором параллелизм добавляется постепенно, пока имеющаяся последовательная программа не превратится в параллельную. Главная задача OpenMP – облегчить написание программ, ориентированных на циклы.

Разработан программный инструмент, который преобразовывает последовательную программу, написанную на языке программирования C/C++, в параллельный код с помощью директив OpenMP. Поскольку наиболее ресурсоемким является выполнение циклов, основное внимание программа уделяется распараллеливанию циклов. На вид параллельных циклов накладываются достаточно жёсткие ограничения. В частности, предполагается, что программа не должна зависеть от того, какой именно поток и какую итерацию параллельного цикла она выполнит. Если тело цикла не удовлетворяет условиям распараллеливания, то пользователю предоставляется описание причины, не позволившей распараллелить цикл.

Таким образом, последовательная программа проходит следующие этапы обработки:

- лексический и синтаксический анализ;
- семантический анализ (проверка ограничений на циклы);

- генерация кода (вставка директив OpenMP) или выдача сообщения об ошибке (вставка комментария о причине, по которой невозвожно распараллелить цикл).

Полученный программный код можно сразу использовать, ничего не корректируя, и при этом, видя, почему тот или иной цикл нельзя распараллелить. Программист может при необходимости отредактировать сгенерированный параллельный код.

В разработанной программе есть возможность задать некоторые настройки, влияющие на те или иные директивы OpenMP: задать число потоков, определить, как будут распределены между ними итерации. Все преобразования исходного кода в параллельный код осуществляются посредством лексического и синтаксического анализаторов, сгенерированных с помощью ANTLR.

ОТ ПОСТАНОВКИ ЗАДАЧИ К ПАРАЛЛЕЛЬНОЙ ПРОГРАММЕ

Предположим, Вы пишете задание на поиск слова в файлах в следующем виде:

```
use boostthread library;
number of threads = 4;
search for "luck";
in "D:\CP\DATA";
print filenames and time;
```

В результате вы автоматически получаете параллельную программу для решения данной задачи. Опишем, как мы сделали это возможным.

Всего в мире по статистике придумано более восьми тысяч языков программирования. Каждый год их число увеличивается. Некоторые языки использует лишь узкая группа специалистов, другие получают широкое распространение. В основном наиболее популярными языками являются те, которые позволяют решать множество различных типов задач. Однако, будучи универсальными, они могут оказаться и более сложными в изучении. Поэтому в некоторых ситуациях при разработке программ для решения конкретных узких задач может возникнуть необходимость в разработке собственного специализированного языка программирования.

В данной работе предложен язык, предназначенный для формализации задач поиска, и разработана его грамматика. Данный раздел как раз и начинается с фрагмента кода на этом языке. Создан программный инструмент, который осуществляет перевод программы, написанной на авторском языке, на язык программирования C++. Результирующая программа имеет характеристики:

- является многопоточной;
- используется модель делегирования;

- может быть реализована в одном из трех вариантов: OpenMP, Boost.Thread и TinyThread++.

Про технологию OpenMP было сказано выше. Boost – это набор бесплатных портативных библиотек. Все библиотеки хорошо взаимодействуют со стандартными библиотеками C++, могут использоваться в широком спектре приложений и помогать в решений множества различных задач. Библиотека Boost.Thread позволяет использовать несколько потоков для выполнения некоторого участка кода. Эта библиотека предоставляет классы и функции для того чтобы управлять самими потоками, а также позволяет синхронизировать данные между потоками. Библиотека TinyThread++ – это очень маленькая портативная реализация базовых классов потоков C++. Основная её задача – сделать разработку многопоточных C++-приложений как можно проще.

Разработан удобный графический интерфейс пользователя, дана возможность задавать параметры поиска, например, количество потоков и используемое средство распараллеливания. Также добавлена возможность предлагать пользователю наиболее эффективные параметры для запуска поиска, выявленные на основании предыдущих экспериментов.

Для получения многопоточных реализаций применяются реализованные на C++ программы-шаблоны поиска. Это позволяет сделать свой язык программирования очень простым: пользователю надо всего лишь указать параметры для поиска (используемую библиотеку, количество потоков, ключевое слово для поиска, файлы, в которых необходимо осуществлять поиск, а также вид возвращаемого результата). Весь остальной код генерируется с использованием шаблонов.

Были проведены эксперименты, оценивающие эффективность библиотек в различных условиях. На основании полученной статистики добавлен функционал для оптимизации входных параметров программы, задаваемых пользователем.

ЗАКЛЮЧЕНИЕ

На сегодняшний день существует ряд систем автоматического распараллеливания. Подавляющее большинство этих систем являются дорогими коммерческими продуктами. Мы лишены даже возможности провести их сравнительный анализ.

В данной работе мы сделали первый шаг по пути «автоматизации распараллеливания программ». Успешный или нет – судить читателю. Мы собираемся идти по нему дальше, к осуществлению «мечты об автоматическом преобразовании хорошего последовательного кода в эффективный параллельный код».