ЯДРО ИНТЕРНЕТ-СЕРВИСА АВТОМАТИЧЕСКОГО РАСПАРАЛЛЕЛИВАНИЯ ПРОГРАММ НА УРОВНЕ ИСХОДНОГО КОДА

Д.В. Курганов, С.А. Немнюгин

В настоящее время все большее число экспериментов и математических расчетов выполняется с использованием электронных вычислительных систем. Научному работнику для организации расчетов на вычислительной машине в первую очередь необходимо составить алгоритм производимых вычислений на понятном для ЭВМ языке. С появлением параллельных вычислительных архитектур стало возможным ускорение вычислений в несколько раз, но в тоже время возникли дополнительные трудности с использованием подобных архитектур, так как реализация приложений под них требует специальных знаний и дополнительных усилий.

Учитывая, что автоматическому распараллеливанию поддается лишь узкий класс алгоритмов, для эффективного распараллеливания так или иначе необходимо вмешательство программиста, поэтому было бы удобно иметь инструмент, который будет брать на себя рутинную часть работы при распараллеливании, оставляя за программистом конечное преобразование кода. Целью работы является создание ядра интернет-сервиса автоматической обработки исходного кода последовательной вычислительной задачи с целью получения на выходе кода, снабженного специальными комментариями и прагмами OpenMP перед циклами «for» для облегчения преобразования последовательной программы в параллельную. Данный проект выполняет анализ OpenMP ограничений на циклы и в случае отсутствия таких ограничений вставляет OpenMP прагму перед циклом, предоставляя программисту право судить об эффективности распараллеливания данного цикла. В будущем возможно расширение функциональности алгоритма на выявление других возможностей распараллеливания. Ранее был представлен доклад[2], в котором подробно была описана архитектура веб-интерфейса данного сервиса. Ядро сервиса разрабатывается на языке C/C++[1]. Основное внимание уделяется простоте в использовании данной системы со стороны пользователя. Таким образом, основной задачей ядра интернет-сервиса является нахождение циклов, удовлетворяющих условиям распараллеливания для OpenMP[3]:

- переменная цикла должна иметь тип signed integer;
- операция сравнения должна иметь следующий формат: переменная_цикла <, <=, >, >= инвариант_цикла_целого_типа;
- третье выражение (или инкрементная часть цикла for) должно являться либо целочисленным сложением, либо целочисленным вычитанием;
- если используется операция сравнения < или <=, переменная цикла должна увеличиваться при каждой итерации, а при использовании операции > или >= переменная цикла должна уменьшаться;
- не разрешены переходы из цикла, за исключением оператора exit, который завершает работу всего приложения. Если используются операторы goto или break, они должны приводить к переходам внутри цикла, а не вне его;
- должна отсутствовать цикловая зависимость по данным, а также чтение и запись в общие для потоков ячейки памяти, т.к. это приводит к “гонкам за данными” и в последствии к недетерминированному результату.

Актуальность данной работы в том, что в настоящее время прикладному программисту приходится тратить много времени для ручной обработки исходного кода последовательной программы с целью преобразования ее к параллельному варианту, данный интернет-сервис позволяет автоматизировать процесс создания такого кода, освобождая программиста от ненужной работы. В частности, из-за массового внедрения многоядерных архитектур и облачных технологий появилась проблема распараллеливания давно и успешно использующихся программных пакетов. Разрабатываемый интернет-сервис позволяет ускорить процесс преобразования программного кода этих пакетов.

Схематически программу можно представить как совокупность нескольких модулей: внешний анализатор, внутренний анализатор, массив деревьев разбора, модуль поиска зависимостей данных, модуль вставки прагм и генератор таблицы переходов парсера. Взаимодействие модулей показано на рисунке.

Файл с исходным кодом на языке C/C++ поступает на вход внешнего анализатора, который рекурсивно осуществляет поиск циклов for, в том числе и вложенные циклы, выделяет тело цикла и разбивает его на отдельные выражения. Также на этом этапе происходит выделение итерационной переменной. Для циклов, чье определение удовлетворяет условиям, внешний анализатор передает управление на внутренний анализатор.

Внутренний анализатор состоит из двух компонентов: лексического и синтаксического анализаторов. Лексический анализатор преобразовывает поток токенов в поток лексем, при этом выделяя итерационную переменную в отдельную лексему. Такой подход облегчает работу синтаксического анализатора а также поиск изменения итерационной переменной и зависимостей данных. Синтаксический анализатор построен по типу

нисходящего LL(1) парсера. Основная функция синтаксического анализатора заключается в построении дерева разбора на основе потока лексем, полученных на выходе лексического анализатора.

После того как исходный код программы полностью обработан и дерево циклов построено, управление передается на модуль вставки прагм. Данный модуль записывает новый файл, который представляет собой копию обрабатываемого файла с вставленными перед циклами for закомментированными прагмами или специальными комментариями. Перед циклами, помеченными как удовлетворяющие условиям распараллеливания, вставляются прагмы #pragma omp parallel for, оформленные в виде комментариев, так как окончательное распараллеливание осуществляется вручную и программист сам выбирает какие из предложенных циклов следует распараллелить. В обратном случае ставится специальный комментарий, указывающий на возможные итерационные зависимости или другие замечания. В настоящее время предусмотрено 5 видов предупреждений:

- “iteration variable must be single”;
- “'break' are found in the loop body”;
- “iteration variable changes in the loop body”;
- “possible data race”;
- “call of function are found in the loop body”.

Одним из важнейших свойств автоматического распараллеливателя является его расширяемость. Расширяемость обеспечивается дополнительным модулем генерации таблицы синтаксического анализатора. Решение обусловлено тем, что язык C/C++ обладает сложным синтаксисом и учесть все языковые особенности не всегда возможно. Чтобы избежать ручного составления таблицы анализатора в случае расширения шаблона обрабатываемых выражений (при изменении правил грамматики) удобно иметь встроенное средство, для генерации таблиц анализатора по заданным грамматикам.

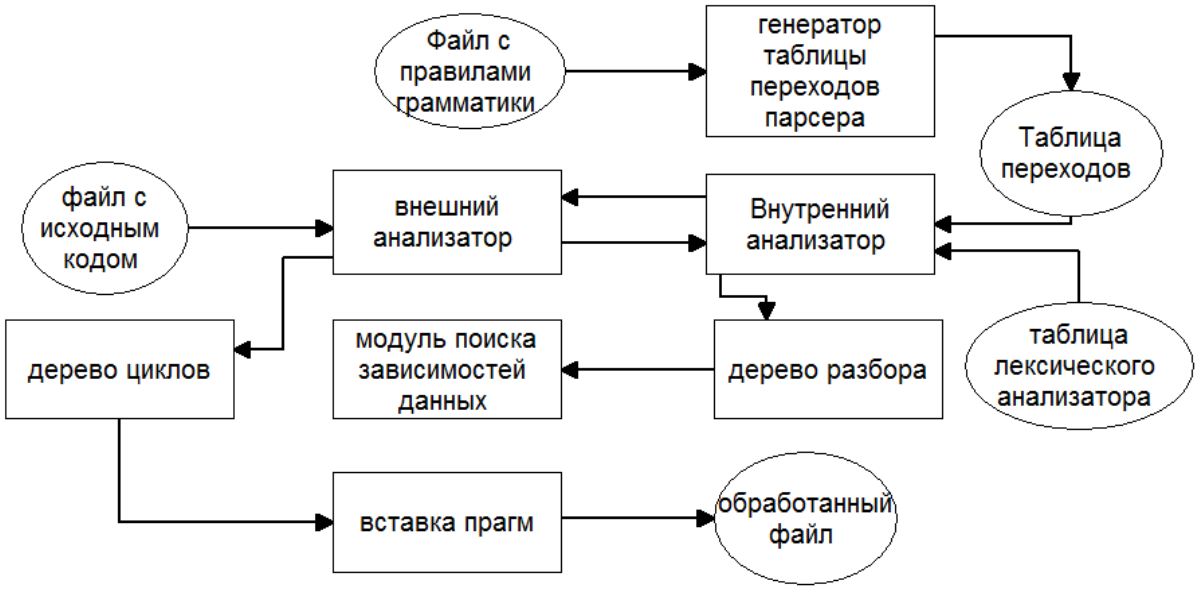

Рис. 1. Структурная схема модулей ядра

Приведем пример работы программы для исходного кода:

```
int main()
{
	int set[245], arr[150];
	int x=0,j=0;
	for(int i=1;i<133;i++)
	{
		x+=2;
		for(int j=1;j<245;j++)
		{
			set[j]=i+j+x;
			if((j%i)==6) break;
		}
	x=i;
	}
	for(int i=1,j =0;i<150;i++,j++)
	{
```

```
            arr[i]=0;
            set[j]+=j;
      }
}
```

После обработки перед каждым циклом программа ставит соответствующий специальный комментарий или OpenMP прагму:

```
int main()
{
      int set[245], arr[150];
      int x=0,j=0;
// #pragma omp parallel for
      for(int i=1;i<133;i++)
      {
            x+=2;
//'break' are found in the loop body;
            for(int j=1;j<245;j++)
            {
                  set[j]=i+j+x;
                  if((j%i)==6) break;
            }
            x=i;
      }
// iteration variable must be single
      for(int i=1, j =0;i<150;i++,j++)
      {
            arr[i]=0;
            set[j]+=j;
      }
}
```

В российском сегменте сети Интернет на данный момент почти отсутствуют аналогичные веб-сервисы, упрощающие разработку параллельного программного обеспечения из имеющихся аналогий можно упомянуть ДВОР (Диалоговый Высокоуровневый Оптимизирующий Распараллеливатель программ)[4]. Данный сервис принимает исходные коды программ написанных только на языке Си, что сильно ограничивает применимость сервиса.

В настоящее время программа обрабатывает исходный код программ, написанных только на языке C/C++, но проект разрабатывается с учетом дальнейшей расширяемости на другие языки и технологии программирования. Среди таких технологий – Intel(R) Cilk™ Plus, а также программирование под сопроцессоры Intel(R) Xeon(R) Phi.

ЛИТЕРАТУРА:

1. Г. Шилдт Полный справочник по C++.-Вильямс,2007.-800с.
2. А.С. Быль, С.А. Немнюгин, Д.А. Пузырев, А.Ю. Петров, Архитектура веб-интерфейса распараллеливания программ на уровне исходного кода, Труды Международной суперкомпьютерной конференции "Научный сервис в сети Интернет: поиск новых решений", 2012, с.551-553.
3. OpenMP Application Program Interface Version 3.0.- http://www.openmp.org/mp-documents/spec30.pdf
4. Б.Я. Штейнберг, А.А. Абрамов, Е.В. Алымова, А.П. Баглий, С.А. Гуда, Д.В. Дубров, Е.Н. Кравченко, Р.И. Морылев, З.Я. Нис, В.В. Петренко, С.В. Полуян, И.С. Скиба, В.Н. Шаповалов, О.Б. Штейнберг, Р.Б. Штейнберг, М. Юрушкин. / Научный сервис в сети Интернет: Труды Всероссийской суперкомпьютерной конференции (20-26 сентября 2010 г., г. Новороссийск). М.: Изд-во МГУ, 2010., с. 71-75