Преобразование последовательных Си-программ для их распараллеливания*

Ю.Г. Зыков[2], Н.А. Катаев[1], А.С. Колганов[1,2]

Институт Прикладной Математики им. М.В. Келдыша РАН [1], Московский Государственный Университет им. М.В. Ломоносова[2]

Параллельные вычислительные технологии (ПаВТ'2018)
Parallel Computational Technologies (PCT'2018)

* Работа поддержана грантами РФФИ 16-07-01067, 17-01-00820 и 18-01-00851.

Современные оптимизирующие компиляторы позволяют задействовать опции, отвечающие за векторизацию и распараллеливание последовательных программ, но их возможности ограничены из-за необходимости трудоемкого статического анализа. Распараллеливание часто требует значительного преобразования программы, в том числе и на уровне исходного кода, а необходимость преобразования определяется возможностями выбранной технологии параллельного программирования. Применение интерактивных систем при распараллеливании программ позволяет найти баланс между сложностью автоматического преобразования и анализа программ и сложностью ручного использования различных технологий программирования пользователем.

Одной из важных оптимизаций является подстановка процедур в Си-программах, а именно, замена вызова процедуры ее непосредственным телом со всеми подставленными аргументами.

```c
#include <stdio.h>
int sum(int *a, int sz) {
  int i, s = 0;
  for (i = 0; i < sz; ++i) {
    s += a[i];
  }
  return s;
}

int main(int argc, char **argv) {
  int m[] = {0, 1, 2, 3, 4, 5, 6, 7};
  if (argc < 2) {
    #pragma spf transform inline
    printf("%d\n", sum(m, sizeof(m) / sizeof(*m)));
  }
  return 0;
}
```

⇩

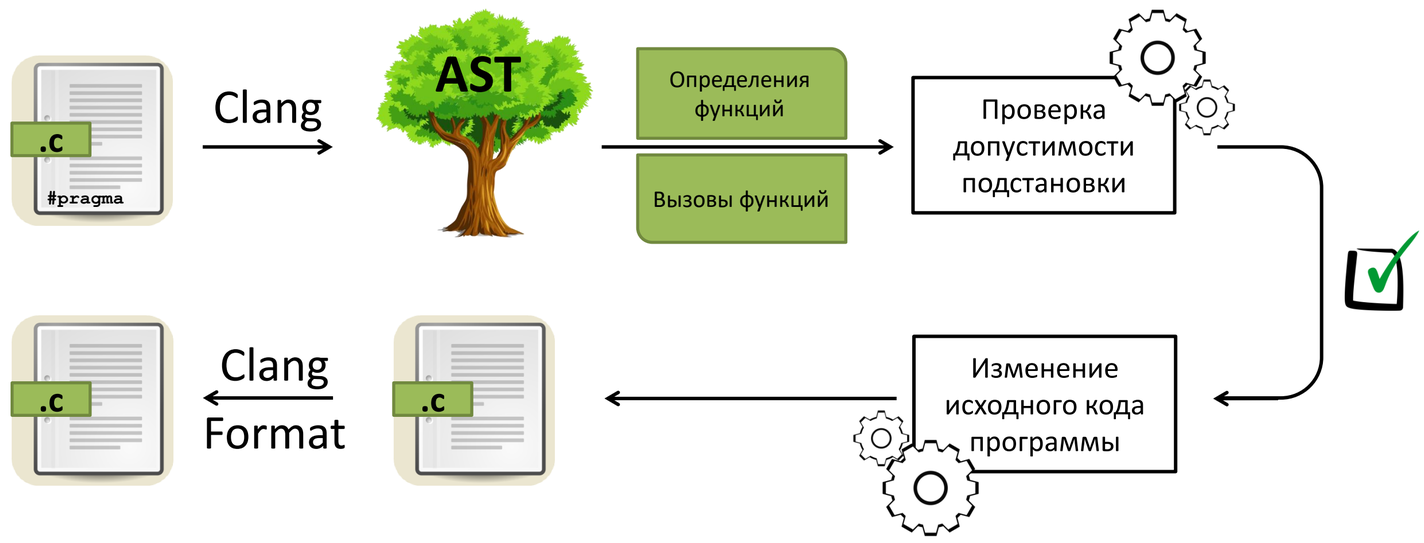
```c
...
int main(int argc, char **argv) {
  int m[] = {0, 1, 2, 3, 4, 5, 6, 7};
  if (argc < 2)
  {
    int r;
    {
      int *a = m;
      int sz = sizeof(m) / sizeof(*m);
      int i, s = 0;
      for (i = 0; i < sz; ++i) {
        s += a[i];
      }
      r = s;
      goto L1;
    }
L1:
    printf("%d\n", r);
  }
  return 0;
}
```

Данная оптимизация позволяет при выполнении программы снизить накладные расходы, связанные с вызовом процедуры, а на этапе статического анализа и компиляции делает возможным применение многих других оптимизаций, в том числе и распараллеливающих, без фактического межпроцедурного анализа.

Применение данной оптимизации на уровне исходного кода позволит системе САПФОР определять более эффективные схемы распараллеливания пользовательской программы.

Модуль для системы САПФОР был реализован на языке C++ с применением инфраструктуры LLVM и Clang и проверен на нескольких тестах из набора NAS Parallel Benchmarks.

Подстановка функций



Подстановка функций не осуществляется, если функция:

- определена вне пользовательских файлов;
- является вариативной;
- является рекурсивной;
- функция имеет глобальные зависимости от вложенно-определенных объектов (например, структура в структуре в структуре);
- имеет глобальные зависимости от объектов, статических или определенных в файле, отличном от определения функции, за исключением подстановок в этом самом файле;
- имеет глобальные зависимости от объектов, если в месте вызова эти объекты будут "скрыты".

Автоматизация распараллеливания последовательных программ

Автоматизация распараллеливания программ вызывает трудности не только из-за ограниченных возможностей статического анализа. Как и оптимизации, осуществляемые современными компиляторами, распараллеливание часто требует значительного преобразования программы. При этом программа изменяется в том числе и на уровне исходного кода, а необходимость преобразования определяется возможностями выбранной технологии параллельного программирования.

Например, стандарт OpenMP обеспечивает распараллеливание циклов, записанных только в каноническом виде. Конструкции DVMH языков позволяют отобразить многомерное гнездо циклов на решетку процессоров, при этом циклы в отображаемом гнезде должны быть тесно вложенными. Аналогично, конструкция collapse стандарта OpenMP требует тесной вложенности объединяемых циклов.

В случае использования ускорителей в модели DVMH, выполняемый на них код может вызвать только такие процедуры, которые не вырабатывают побочный эффект. Одним из способов устранения процедур с побочным эффектом могла бы быть их подстановка в исходный код, которую можно выполнять либо в коде исходной программы, либо в процессе ее конвертации DVMH компилятором. Стоит отметить, что в последнем случае конвертация выполняется на уровне исходного кода и сконвертированная программа остается записанной на языке высокого уровня (Си, Фортран), соответственно подстановка процедур также должна быть выполнена на уровне исходного кода.

Система САПФОР ориентирована на инкрементальное, итерационное, интерактивное распараллеливание последовательных программ на многоядерные кластеры с ускорителями и рассматривает процесс распараллеливания программы, как поиск оптимизационной последовательности проходов анализа и преобразования, обеспечивающей переход от последовательной версии программы к эффективной параллельной версии.

```c
void updateBRow(int idx) {
  for (int i = 1; i < L-1; i++)
    B[idx][i] = (A[idx-1][i] + A[idx][i-1]
      + A[idx][i+1] + A[idx+1][i]) / 4.0;
}
```

•••

```c
/* Iteration process of Jacoby 2D*/
for (int it = 1; it <= ITMAX; it++) {
  ...
  for (int i = 1; i < L - 1; i++)
    updateBRow(i);
  ...
}
```

Фрагмент исходного кода

⇨

```c
/* Iteration process of Jacoby 2D*/
for (int it = 1; it <= ITMAX; it++) {
  ...
  for (int i = 1; i < L - 1; i++)
  /* updateBRow(i) is inlined below */
  {
    int idx0 = i;
    for (int i = 1; i < L-1; i++) {
      B[idx0][i] = (A[idx0-1][i] + A[idx0][i-1]
        + A[idx0][i+1] + A[idx0+1][i]) / 4.0;
    }
  }
  ...
}
```

Результат применения подстановки функций

⇨

```c
/* Iteration process of Jacoby 2D*/
for (int it = 1; it <= ITMAX ; it++) {
  ...
  #pragma dvm parallel([i][j] on B[i][j]) shadow_renew(A)
  for (int i = 1; i < L - 1; i++)
  /* updateBRow(i) is inlined below */
  {
    for (int i0 = 1; i0 < L-1; i0++) {
      B[i][i0] = (A[i-1][i0] + A[i][i0-1]
        + A[i][i0+1] + A[i+1][i0]) / 4.0;
    }
  }
  ...
}
```

Распараллеленный фрагмент кода после подстановки переменных

Отсутствие тесно вложенного гнезда из двух циклов:

- препятствует распараллеливанию вычислений по каждому измерению массива в случае использования DVMH;
- снижает эффективность распараллеливания из-за невозможности использования collapse в случае использования OpenMP.

⟶

Распараллеливание становится возможным после применения последовательности преобразований:

- подстановка функций;
- подстановка и переименование переменных.

С другими работами связанными с САПФОР и DVM можно ознакомится на сайте ***http://dvm-system.org***