Московский физико-технический институт (национальный исследовательский университет)
Физтех-школа радиотехники и компьютерных технологий
Кафедра информатики и вычислительной техники

Реализация поддержки выражений в директивах OpenMP в оптимизирующем компиляторе для архитектуры Эльбрус

Выпускная квалификационная работа
(бакалаврская работа)

Студент: Сущенко А.А., 813 группа
Научный руководитель: Горелов М.А.

Москва, 2022

Введение

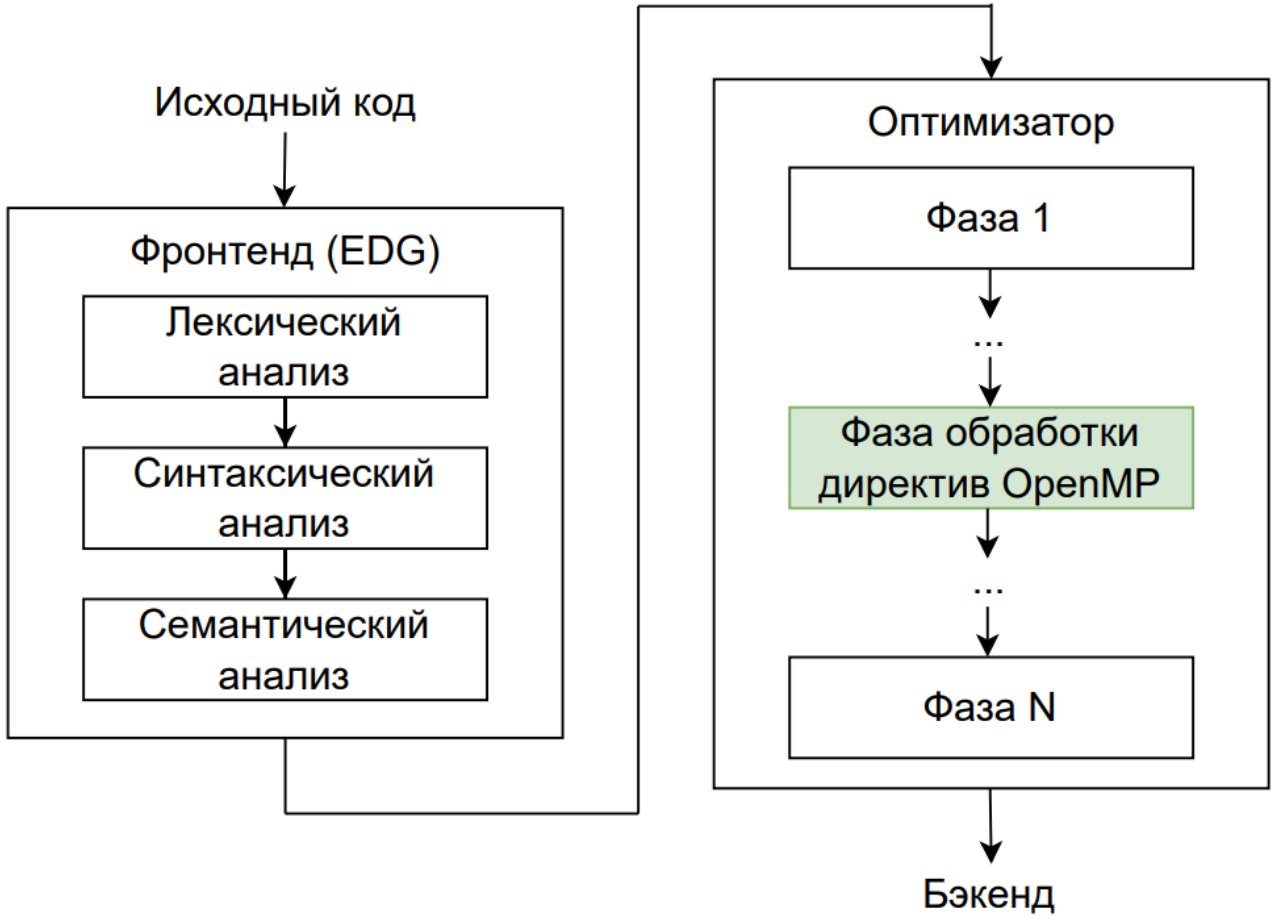
- OpenMP — API для распараллеливания программ на языках C, C++ и Fortran

- OpenMP состоит из набора директив и библиотеки *libomp*

- Директивы OpenMP на языках C и C++:

```
#pragma omp [директива] [опция1, опция2, ...]
```

Обработка директив OpenMP



- Фронтенд EDG не обрабатывает директивы OpenMP
- Обработка директив OpenMP происходит в оптимизаторе компилятора

Описание проблемы

- Фаза обработки OpenMP поддерживает в директивах только тривиальные выражения

- Пользователям приходится проводить вычисления выражений перед прагмой в коде

Неподдерживаемый пример

```
#pragma omp parallel if (b + c)
```

Поддерживаемый пример

```
int a = b + c;
#pragma omp parallel if (a)
```

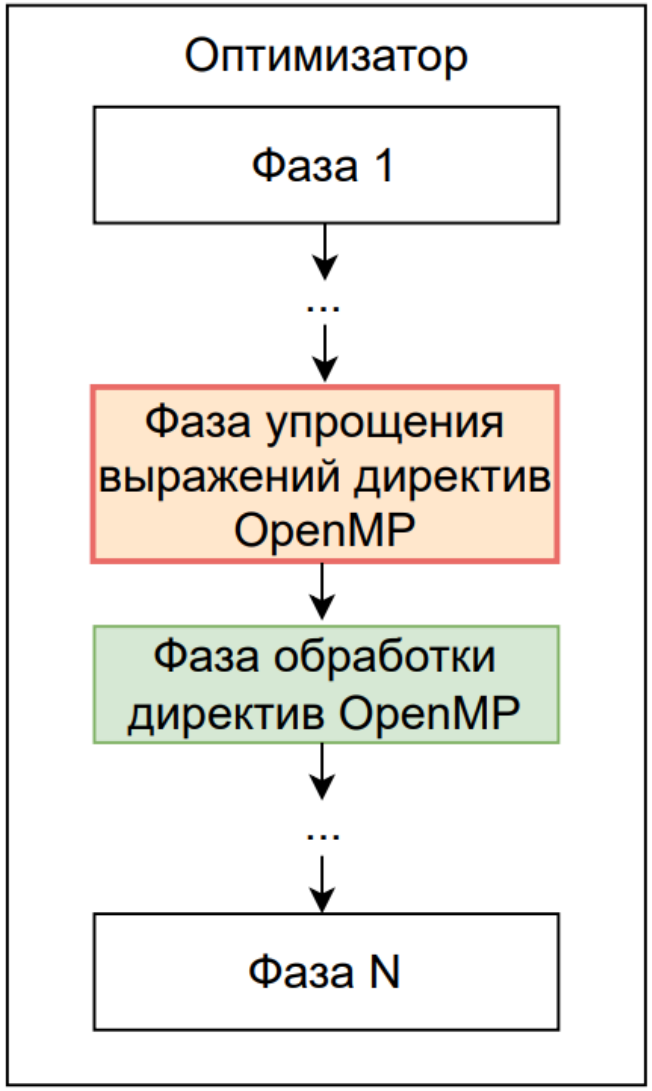
- В ряде задач SPEC CPU также присутствуют неподдерживаемые конструкции

Необходимо скорректировать исходный код

Нельзя публиковать результаты производительности данных задач

Цель работы

Поддержать выражения интегрального типа внутри директив OpenMP согласно стандарту версии 4.0 в оптимизирующем компиляторе путем реализации фазы, которая производит построение внутреннего представления выражения



Задачи работы

- Реализовать фазу упрощения выражений в оптимизаторе компилятора
- Поддержать в фазе упрощения:
  - Элементарные операторы
  - Скобочные выражения
  - Операторы преобразования типов
  - Функции из библиотеки OpenMP
- Реализовать диагностику пользовательских ошибок при обработке выражений

Построение кода вычисления выражения

1. Обход операций CALL(pragma) с директивой omp для проверки выражений внутри неё

parallel
teams
task
cancel

2. Поиск стартового слова, после которого следует выражение

if(
num_teams(
num_threads(
thread_limit(

3. Упрощение выражения после стартового слова

- Обход операторов в порядке приоритета
- Построение операций в промежуточном представлении для вычисления значения выражения

Итог: Замена всего выражения на одну переменную ***__new_obj_elim_expr_N***

Было

```
CALL … {“omp parallel if ( a * b – (c % (long int)d) + (k) )”}
```

Стало

```
Операции вычисления значения выражения
WRITE loc:__new_obj_elim_expr_5 <- результат
CALL … {“omp parallel if (__new_obj_elim_expr_5)”}
```

Пример обработки простого выражения

До применения фазы упрощения

```
o1. CALL … {“omp parallel if (a + b < c)”}
```

После применения фазы упрощения

```
o2. READ     loc:a
o3. READ     loc:b
o4. ADD_I    o2, o3
o5. READ     loc:c
o6. SLT_I    o4, o5 (boolean)
o7. B2I      o6
o8. WRITE    loc:__new_elim_obj_expr_2 <- o7
o1. CALL … {“omp parallel if (__new_elim_obj_expr_2)”}
```

Пример обработки выражения с вызовом функции

До применения фазы упрощения

```
o1. CALL … {“omp parallel if (2 - omp_get_max_threads())”}
```

После применения фазы упрощения

```
o2. CALL      … omp_get_max_threads()
o3. CONST     (sint32)0x2
o4. SUB_I     o2, o3
o5. WRITE     loc:__new_obj_elim_expr_2 <- o4
o1. CALL … {“omp parallel if (__new_elim_obj_expr_2)”}
```

Диагностика пользовательских ошибок

- Необъявленные переменные
- Нарушен баланс скобок
- Неправильное использование оператора

I. Передача исходного выражения в следующие фазы:

```
#pragma omp parallel if (a + 2) abc
```

- Для нашей фазы ошибки отсутствуют => будет заменено выражение внутри прагмы
- Исходное выражения хранится в хэш-таблице для вывода ошибки пользователю

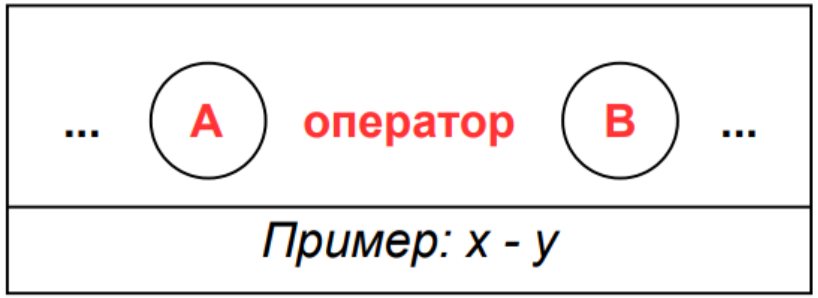
```
lcc: “test.c”, line 5: error: misspelling pragma line “omp parallel
       if(a + 2) abc” (unknown keyword “abc”)
```

Диагностика пользовательских ошибок

II. Диагностика ошибок внутри фазы

1. Проверка целостности скобок при поиске стартового слова

2. Проверка аргументов **A** и/или **B** оператора при его упрощении, в том числе операторов приведения типов:
   - Непустой аргумент (например, « »)
   - Существование аргумента
   - Тип аргумента для некоторых операторов (например, «<<»)



Примеры пользовательских ошибок

Пример №1

```
#pragma omp parallel if (!=2)
```

```
lcc: “test.c”, line 5: error: expected expression before “!=”, input – “if (!=2)”
```

Пример №2

```
#pragma omp parallel if (s!=2))
```

```
lcc: “test.c”, line 5: error: extra bracket after expression “if (s!=2)”
```

Пример №3

```
#pragma omp parallel if (omp_get_max_threads < 2)
```

```
lcc: “test.c”, line 5: error: “omp_get_max_threads” undeclared,
       clause “if (omp_get_max_threads < 2)”
```

Результаты

1. Реализована поддержка выражений в директивах OpenMP путем создания фазы упрощения выражений в оптимизирующем компиляторе
2. Фаза была протестирована и отлажена на задачах SPEC CPU (SPEC2017, SPEC2012) и на коротких нацеленных тестах
3. Реализованная фаза позволила устранить необходимость изменения исходного кода задач SPEC CPU с выражениями в директивах OpenMP
4. Реализация была успешно внедрена в стабильную версию оптимизирующего компилятора