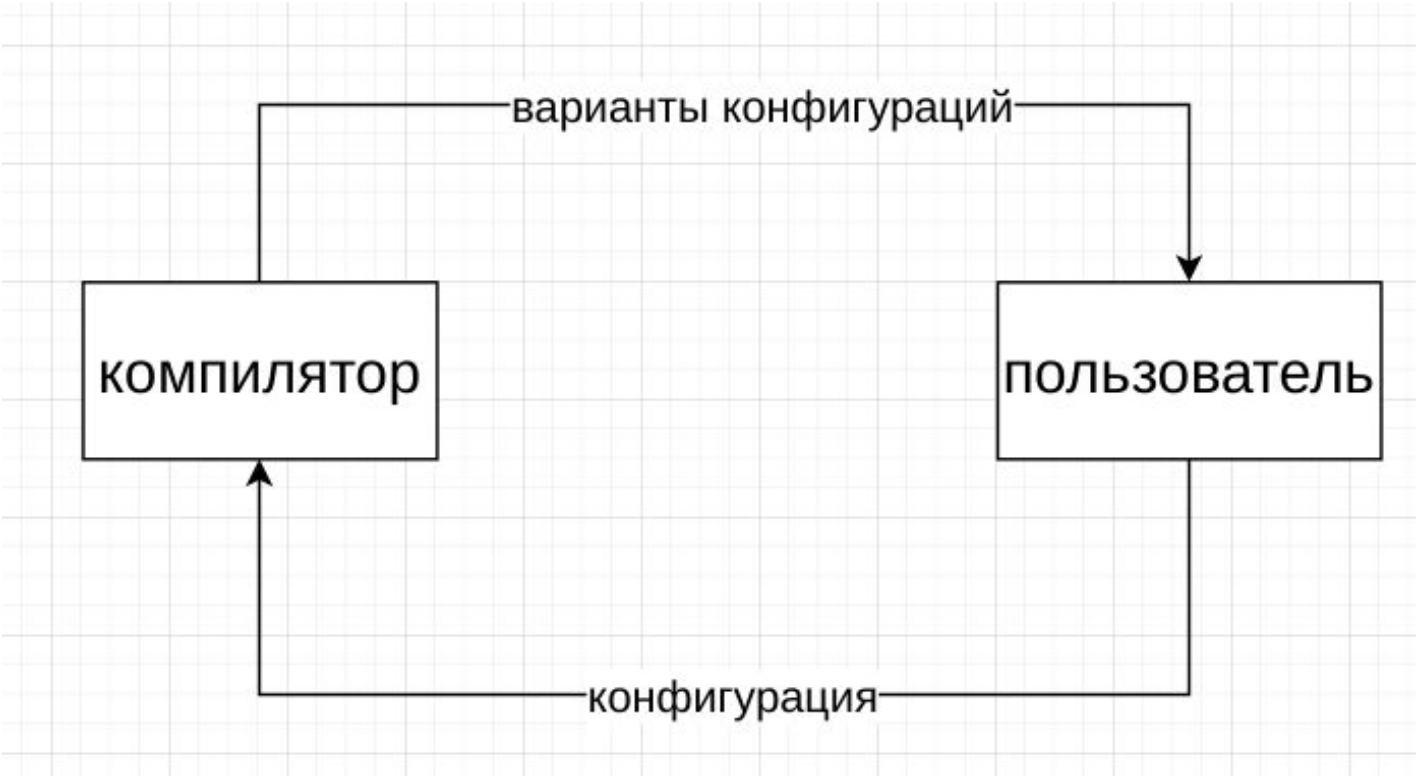
XXXIV Зимняя школа по параллельному программированию 2020

Расширение компилятора LuNA возможностью подключать модули распределения ресурсов

Выполнил: Ижицкий Р. Л.
Руководитель: Перепелкин В. А.
31.01.2020

Схема


варианты конфигураций
компилятор
пользователь
конфигурация

Цель проекта

- Обеспечить возможность настраивания автоматического отображения фрагментов LuNA программы на ресурсы

Задачи на ЗШ2020

- Обнаружить фрагменты, требующие распределения
- Выдать множество вариантов распределения фрагментов
- Применить заданную конфигурацию
- Возможность накопления модулей

Локатор

- CyclicLocator(n): n – номер узла, где находится (будет находиться) блок
- По умолчанию все находится на 0 узле, если в коде не указано обратное

```
// BI_EXEC: cf d: init(val, x[6]);
BlockRetStatus block_6(CF &self)
{
    if (self.migrate(CyclicLocator(0))) {
        return MIGRATE;
    }
    …
}
```

final.cpp

Задачи ЗШ2020

- Выдать множество рекомендаций для локатора из компилятора в Looper (какой локатор использовать вместо локатора по умолчанию)
- Обработать обратную связь от Looper в компиляторе

1 Генерация множества рекомендаций

- --gen-conf=relative_path_to_config

```
0 default hash first_increase first_decrease second_increase second_decrease
1 default hash
2 default hash first_increase first_decrease
3 default hash
4 default hash first_increase first_decrease second_increase second_decrease
5 default hash
6 default hash
```

config_file

1.1 Исходный код

- Введено правило idx_count

```
sub display(name y, int idx)
{
    cf c: iprint(y[idx*2][6]) @ {
        request y[idx*2][6];
    };
} @ {
    idx_count y=>2;
}
```

.fa file

1.2 JSON-представление

- Обработан JSON для **exec**, for, if, while, **name** (только для struct), df

```
"args": [{"type": "name", "id": "y"},],
"body": [{"type": "exec", "id": ["C"], "code": "iprint", ...}],
"rules": [
    {
        "property": "idx_count",
        "id": ["Y"],
        "expr": {"type": "iconst", "value": 2}
    }
],
```

.ja file

1.3 Обработка JSON-представления

- Соответствие всех блоков уникальным идентификаторам

```
>> ID = y  UID = 0  COUNT = 2
>> ID = c  UID = 1
>> ID = x  UID = 2  COUNT = 1
>> ID = d  UID = 3
>> ID = z  UID = 4  COUNT = 2
>> ID = a  UID = 5
>> ID = b  UID = 6
```

debug log

ID / Name	Unique ID	Arg number
a	5	0
b	6	0
c	1	0
d	3	0
x	2	1
y	0	2
z	4	2

1.4 Генерация рекомендаций

- В компилятор внедрен существующий модуль

```
all_locators = {
    'default': [default_locator, 0],
    'hash': [hash_locator, 0],
    'first_increase': [first_index_increase_locator, 1],
    'first_decrease': [first_index_decrease_locator, 1],
    'second_increase': [second_index_increase_locator, 2],
    'second_decrease': [second_index_decrease_locator, 2],
    'third_increase': [third_index_increase_locator, 3],
    'third_decrease': [third_index_increase_locator, 3]
}
```

locator.py

1 Генерация множества рекомендаций

- --gen-conf=relative_path_to_config

```
0 default hash first_increase first_decrease second_increase second_decrease
1 default hash
2 default hash first_increase first_decrease
3 default hash
4 default hash first_increase first_decrease second_increase second_decrease
5 default hash
6 default hash
```

config_file

2.1 Использование рекомендаций

- --use-conf=relative_path_to_config

```
0 second_increase
1 default
2 first_decrease
3 default
4 second_decrease
5 default
6 default
```

config_file

2.2 Повторная генерация unique ids

- Соответствие всех блоков уникальным идентификаторам

```
> id = c  uid = 1  count = None  used_locator = default
> id = y  uid = 0  count = 2  used_locator = second_increase
> id = d  uid = 3  count = None  used_locator = default
> id = x  uid = 2  count = 1  used_locator = first_decrease
> id = b  uid = 6  count = None  used_locator = default
> id = z  uid = 4  count = 2  used_locator = second_decrease
```

debug log

2.3 Генерация рекомендаций

- В компилятор внедрен существующий модуль

```
def the_locator(loc_name, id):
    global all_locators
    loc, _ = all_locators.get(loc_name)
    return loc(id)
```

locator.py

```
def locator(id, scope, rules_scope=None):
    if rules_scope['parent'] is None:
        return 'CyclicLocator(%s)' %
            value_int(the_locator(loc_name, id), scope)
```

компилятор

2.4 Генерация скомпилированного кода

- x – first_decrease, z – second_increase

```
// req_count x[6]=2
{         DF posted=self.wait(self.id(0)[6]);
     self.post(self.id(0)[6], posted, CyclicLocator(static_cast<int>((0)-(6))), 2);
}

…

// request z[10][6]
self.request(self.id(0)[10][6], CyclicLocator(6));
```

final.cpp

0.72 Генерация stealable

- В компилятор внедрен существующий модуль

```
// BI_EXEC: cf a: init(7, x[4]);
BlockRetStatus block_2(CF &self)
{
    if (self.check_steal()) { return STEAL; }
```

final.cpp

```
def gen_stealable(scope, ja):
    if flags[0] == "stealable:
        return '\t' + 'if (self.check_steal()) {' + '\n\t\t' + 'return
STEAL;' + '\n\t' + '}' + '\n\n'
```

компилятор

Заключение

- В компилятор внедрен модуль для локатора
  - Генерация/считывание конфигурационного файла в зависимости от флага компилятору
- Генерация кода для stealable

Спасибо за внимание