Трансляция функциональных языков в императивные

Дипломная работа Климова Ивана,
461 группа

Научный руководитель: магистр информационных технологий Григорьев С.В.
Рецензент: ведущий архитектор, ООО "Belkasoft" Тимофеев Н. М.

Предметная область

- ЯВУ
  - Языки для работы с графическими процессорами
  - Функциональные языки
- Массовый параллелизм на GPGPU
- Brahma.FSharp

Постановка задачи

Цель

- Интеграция ЯВУ с вычислениями на GPGPU

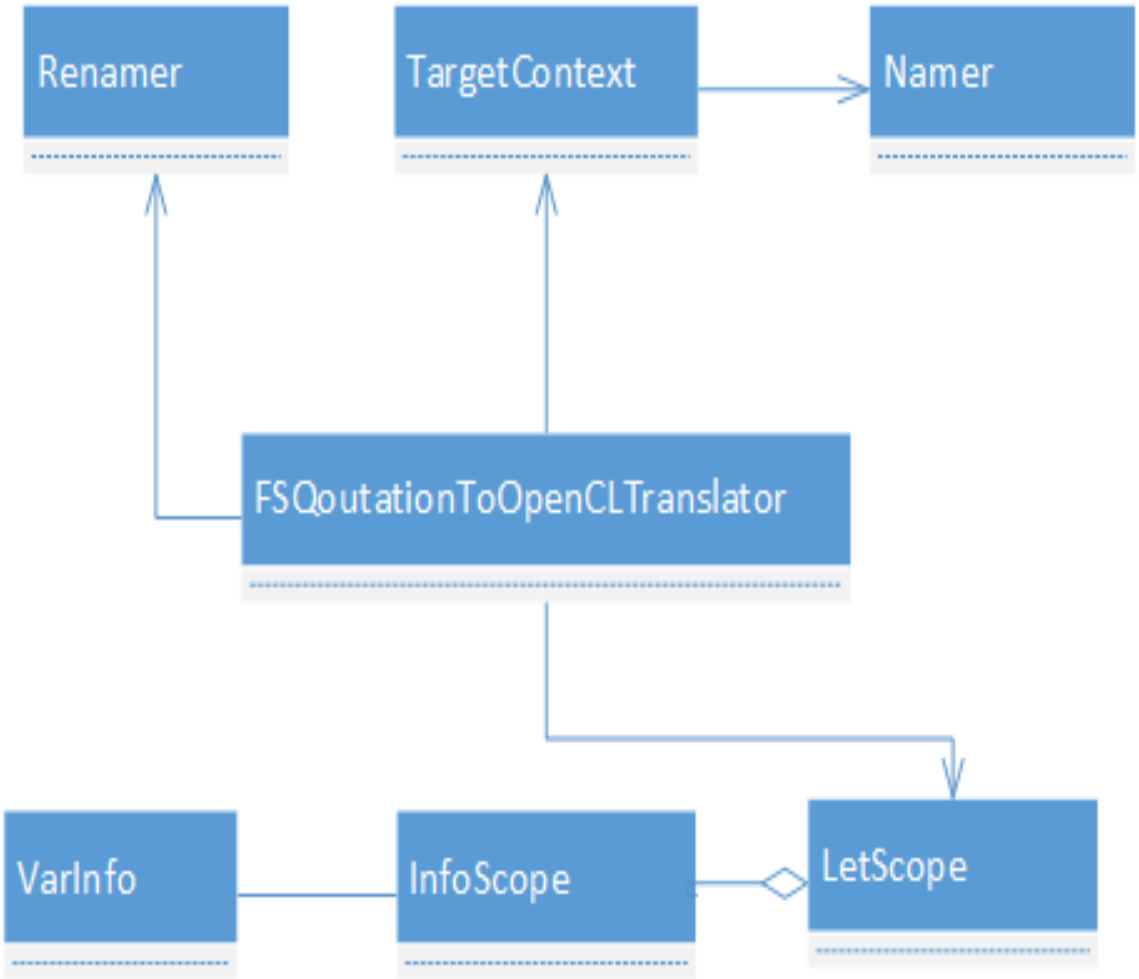
Задачи

- Изучить особенности языка F#
- Разработать шаблоны трансляции в язык OpenCL
- Реализовать поддержку трансляции вложенных функций
- Апробация
  - Применение расширенной библиотеки Brahma.FSharp для EigenCFA.

Аналоги

- Alea.cuBase
  - Из F# в CUDA
- FSCL
  - Из F# в OpenCL

Этап трансляции



Шаблоны трансляции

Шаблон рекурсивного выноса функций.

```
let f x =
    let g y =
        let m k = k - 1
        5 + (m y)
    g x
f 5
```

```
let m k = k - 1
let g y = 5 + (m y)
let f x =
    g x
f 5
```

Шаблоны трансляции

Шаблон рекурсивного переименования вложенных функций.

```
let f x =
    let f y =
        let f k = k - 1
        5 + (f y)
    f x
f 5
```

```
let f1 k = k - 1
let f0 y = 5 + (f1 y)
let f x =
    f0 x
f 5
```

Шаблоны трансляции

Шаблон добавления аргументов во вложенные функции.

```
let f x =
    let k = 7
    let g y =
        k + y
    g x
f 5
```

```
let g k y = k + y
let f x =
    let k = 7
    g k x
f 5
```

Реализация

```
let f x =
    let  y = 2
    let g n =
        n + x + y
    g 4
```

Реализация

```
let f x =
    let  y = 2
    let g n =
        n + x + y
    g 4
```

→

```
let g x y n =
    n + x + y
let f x =
    let y = 2
    g x y 4
```

Реализация

```
let f x =
    let  y = 2
    let g n =
        n + x + y
    g 4
```

→

```
let g x y n =
    n + x + y
let f x =
    let y = 2
    g x y 4
```

→

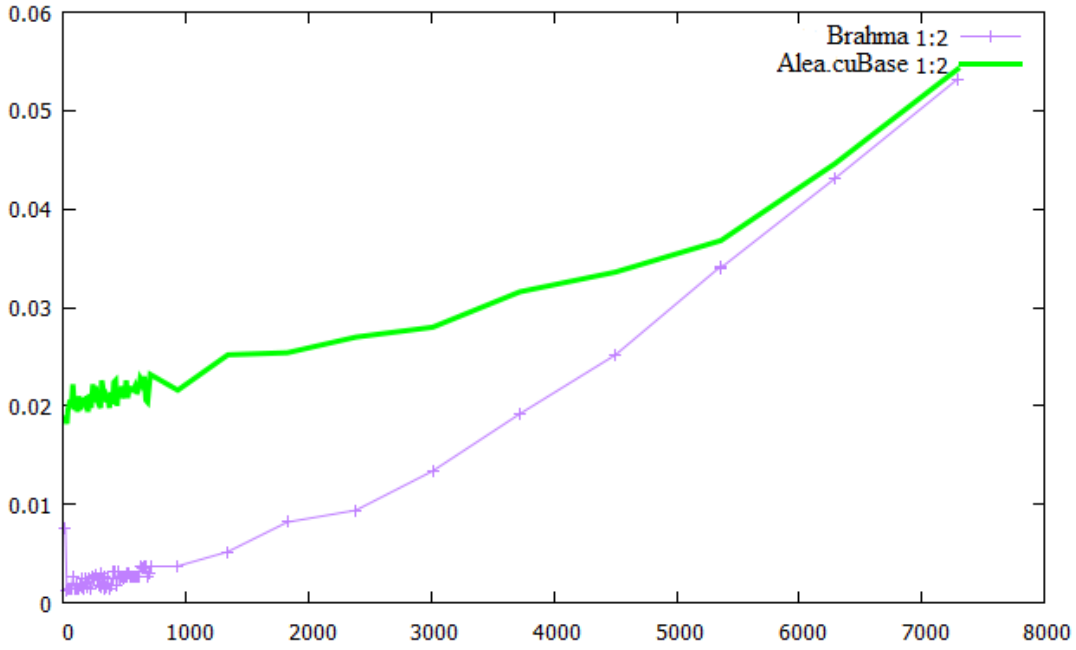
```
int g(int x, int y, int n)
{
    return n + x + y;
}
int f(int x)
{
    int y =  2;
    return g(x, y, 4);
}
```

Алгоритм EigenCFA на F#

```
let qEigenCFA =
        <@ fun (r:_2D)
            (devFun:array<_>) (devArg1:array<_>) (devArg2:array<_>) (devStore:array<_>)
            (devRep:array<_>) devScaleM devScaleCall devScaleLam ->
               let column = r.GlobalID0
               let row = r.GlobalID1
               if(column < devScaleCall && row < 2) then
                    let numCall = column
                    let Argi index =
                        if(index = 0) then devArg1.[numCall]
                        else devArg2.[numCall]
                    let L index = devStore.[devFun.[numCall]*devScaleM + index]
                    let Li index = devStore.[(Argi row)*devScaleM + index]
                    let rowStore row column = devStore.[row*devScaleM + column]
                    let vL j =
                        if(row = 0) then
                            (L j) - 1
                        else
                            (L j) - 1 + devScaleLam
                    for j in 1 .. ((L 0) - 1) do
                        for k in 1 .. ((Li 0) - 1) do
                            let mutable isAdd = 1
                            let addVar = (Li k)
                            for i in 1 .. ((rowStore (vL j) 0) - 1) do
                                if((rowStore (vL j) i) = addVar) then
                                    isAdd <- 0
                            if(isAdd > 0) then
                                devRep.[0] <- devRep.[0] + 1
                                let tail = (rowStore (vL j) 0)
                                devStore.[(vL j)*devScaleM] <- devStore.[(vL j)*devScaleM] + 1
                                devStore.[(vL j)*devScaleM + tail] <- addVar
        @>
```

Сравнение аналогов на EigenCFA

- FSCL
- Alea.cuBase
- Brahma.FSharp



Полученные результаты

- Изучены особенности языка F#
- разработаны шаблоны трансляции в язык OpenCL
- Реализована поддержка трансляции вложенных функций
- Апробация
  - Применение расширенной библиотеки Brahma.FSharp для EigenCFA.