Workflow Builder для библиотеки Brahma.FSharp

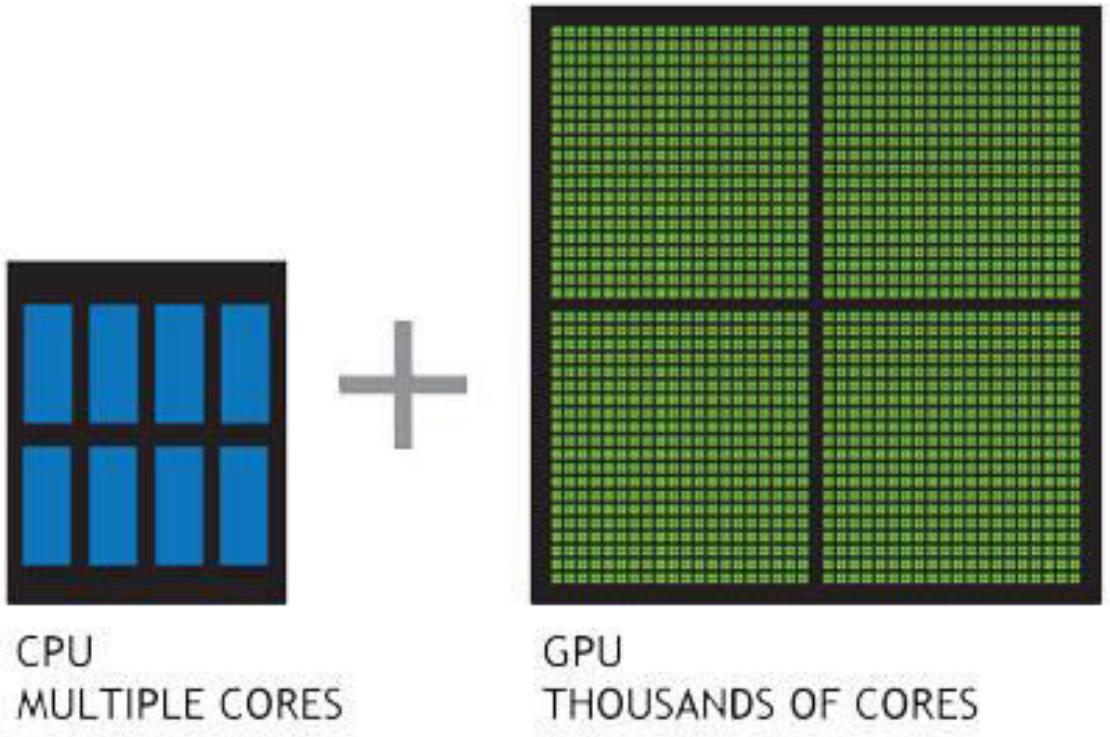
Автор: Васенина Анна Игоревна, 243 группа
Научный руководитель: Григорьев Семён Вячеславович

Санкт-Петербургский государственный университет
Кафедра системного программирования

17 октября 2017 года

Введение

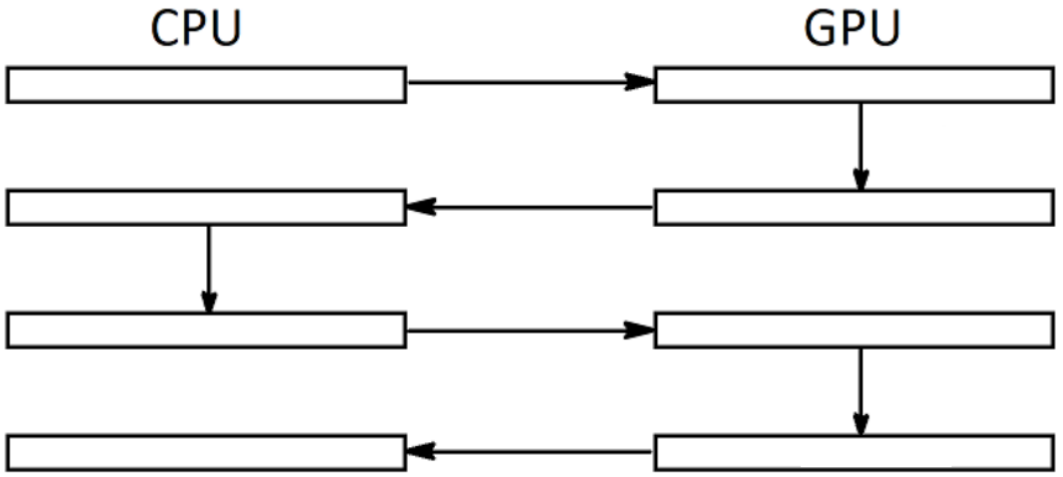
GPGPU (General-purpose computing for graphics processing units, неспециализированные вычисления на графических процессорах) — вычисление неграфических данных на графических процессорах



GPU и CPU (источник: http://www.nvidia.ru/object/gpu-computing-ru.html)

Обзор

Brahma.Fsharp -библиотека на F# для интеграции GPGPU-вычислений по технологии OpenCL.



Используемые технологии:

- Computation expressions
- Монада reader

Цель и задачи

Целью работы является автоматизация манипуляции контекстом при работе на GPU

Задачи:

- Спрятать передачу контекста
- Выполнять операции на GPU без возвращения на CPU до того, как нам потребуется узнать результат

Решение

```
type ReaderM<’context, ‘out> = ‘context -> ‘out

val run: ‘a -> ReaderM<’a,’b> -> ‘b

val constant: ‘c -> ‘a ->‘c

val bind: ReaderM<‘context, ‘out> → f:( ‘out →
                    ReaderM<‘context,‘b>) → ‘context → ‘b
```

Решение

```
type GPUBuilder (actcontext: context) =
 let provider = prov actcontext
 let mutable commandQueue = CQ actcontext
 let length = len actcontext
 let localWorkSize = WS actcontext

 member __.Bind(m, f)    = m >>= f
 member __.Yield (outArr: array<_>) =
    let _ = commandQueue.Add(outArr.ToHostprovider).Finish()
    constant outArr
 member __.Zero = constant None
 member __.Return (outArr: array<_>) =
    let _ = commandQueue.Add(outArr.ToHost provider).Finish()
    commandQueue.Dispose()
    provider.Dispose()
    provider.CloseAllBuffers()
    constant outArr
 member __.Delay(f) = f ()
```

Пример

```
let outerFunc inArr = gpu
    {
     let! e = ArrayGPU.Reverse inArr
     let! f = ArrayGPU.Map <@ fun a -> a + 1 @> e
     yield f
    }

let func1 a = gpu
    {
     let! c = ArrayGPU.Reverse a
     let! d = outerFunc a
     let! g = ArrayGPU.Map2 <@ fun a b -> a + b @> c d
     return g
    }
```

Результаты

1) Реализована возможность выполнять операции на GPU, возвращаясь на CPU только при вызове функции return
2) Реализована неявная передача контекста функциям
3) Проведено тестирование в системе NUnit