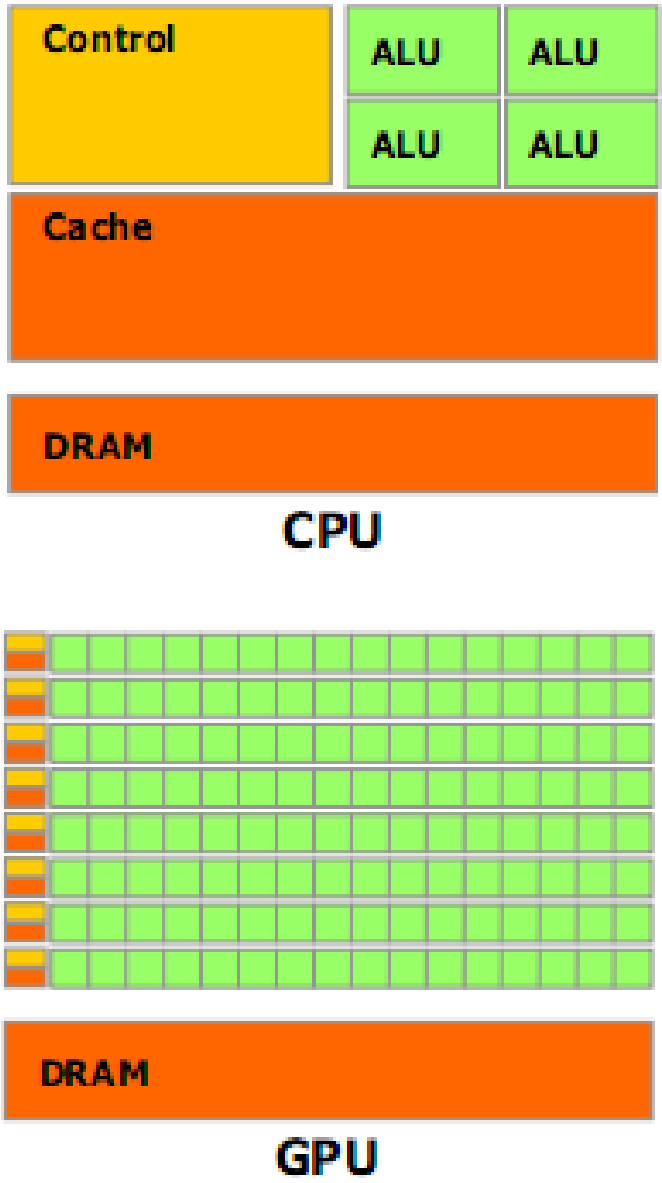
F# OpenCL C Type Provider

Kirill Smirenko, **Semyon Grigorev**

JetBrains Research, Programming Languages and Tools Lab
Saint Petersburg University

September 27, 2018

GPGPU



General purpose computations on grphical processor units

- (Almost) SIMD architecture
- Huge amount of “simple” ALUs on single chip
- May be a good choice for huge data processing

General purpose applications of GPGPU

- Initially for sientific computations
  - Phisics
  - Math
  - Chemistry
- But more and more for applications
  - Finance/Banking
  - Data Analytics and Data Science (Hadoop, Spark ...)
  - Security analytics (log processing)
  - Some “scintific computations” todey are dayle-used applications (bioinformatics, chemistry , . . .)

High level languages and GPGPU

Low-level platforms and languages for GPGPU programming

- NVIDIA CUDA: Cuda C, Cuda Fortran
- **OpenCL: OpenCL C**

High-level platform and languages for applications

- C++
- Pyhon, Haskell, OCaml, ...
- JVM: Java, Scala, ...
- **.NET: C#, F#, ...**

High level languages and GPGPU

Low-level platforms and languages for GPGPU programming

- NVIDIA CUDA: Cuda C, Cuda Fortran
- **OpenCL: OpenCL C**

High-level platform and languages for applications

- C++
- Pyhon, Haskell, OCaml, ...
- JVM: Java, Scala, ...
- **.NET: C#, F#, ...**

Interaction is a problem!

Existing solutions and its problems

- Generative approach (haskell, Alea, etc): good byt what about code reusing?
- Simple drivers (textual, for example) — flexible but not safe.

F# type providers

our focus is .NET, so it is the way to solve our problem Compile-tyme metaprogramming -> design-time features in IDE completion, type informtion, etc

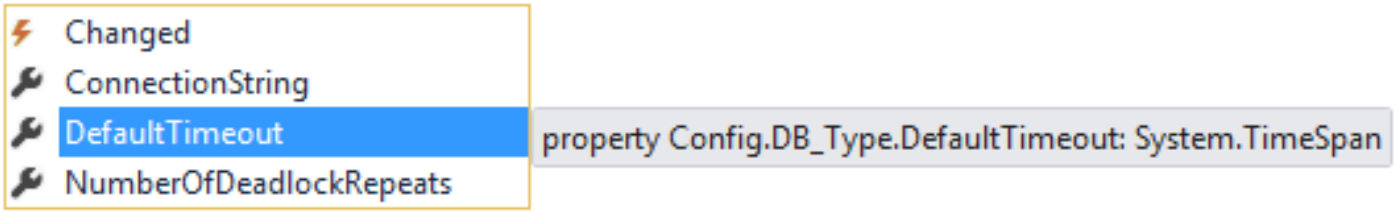
F# type providers

more inside, exaples of R, COM, INI providers

- Config

```
type Config = YamlConfig<"Config.yaml">
let config = Config()

config.DB.
```



Brahma.FSharp

- F# quotations to OpenCL C translator
- Runtime
  - Comand queue
  - Context management
  - Memory management
  - F# aliases for OpenCL-specifc functions

OpenCL C type provider

- Imorove OpenCL C lexer, parser and translator
- Unify kernels on client side
- Improve user exp

Architecture

Diagram

Limitations

- Only (small) subset of OpenCL C
  - h files
  - preprocessor
  - subset of sintax
  - !!!
- Very simple type mapping
- !!!
- !!!

Examples

Screens

Future work

- Imorove OpenCL C support
  - Lexer and parser
  - Translator
  - Types mapping
  - Headers files processing
  - ...
- Unify kernels on client side
  - Currently native Brahma.FSharp's kernel and kernel loaded by type provider are different types
- Improve usability:

Summary

- F# OpenCL C type provider
  - Type-safe integration of existing OpenCL C code in F# applications
  - Prototype with limitations

- Source code on GitHub:
  `https://github.com/YaccConstructor/Brahma.FSharp`
- Package on NuGet:
  `https://www.nuget.org/packages/Brahma.FSharp/`

Contact Information

- Semyon Grigorev: s.v.grigoriev@spbu.ru
- Kirill Smirenko: k.smirenko@gmail.com

- Brahma.FSharp:
  https://github.com/YaccConstructor/Brahma.FSharp

Thakns!