Brahma.FSharp как средство “прозрачного” использования GPGPU в программах на F#

Семён Григорьев

JetBrains Research, лаборатория языковых инструментов
Санкт-Петербургский государственный университет

30.11.2017

План

- GPGPU в F#: но зачем?
  - Есть ли будущее у таких решений?
  - Где их можно применять?
- Почему именно так, а не инече?
  - Уместен ли такой подход?
  - Может можно проще?
- Что под капотом у Brahma.FSharp?

Зачем GPGPU?

- Обработка больших объёмов данных “регклярным” способом
  - Динамический параллелизм...
  - Анализ сетей (социальных, интернет и т.д.)

Зачем GPGPU?

- Обработка больших объёмов данных “регклярным” способом
  - Динамический параллелизм...
  - Анализ сетей (социальных, интернет и т.д.)
- Почему именно так, а не инече?
  - “Надёжность”
  - “Прозрачность”/гомогенность
- Что под капотом у Brahma.FSharp?

Массовый параллелизм

- FSCL
- Alea GPU

Доступ к GPGPU из .NET

- Средства программирования видеопроцессоров в .NET
  - Alea GPU (CUDA)
  - Brahma.FSharp (OpenCL)
  - FSCL (OpenCL)
- Средства запуска CUDA-кода из ЯВУ:
  - CUSP
  - ManagedCuda
- “Низкоуровневые драйвера”
  - OpenCL.NET (`http://openclnet.codeplex.com/`)
  - CUDA.NET

Провайдеры типов

- Функция построения типа по пользовательскому контексту
- Преимущества перед кодогенерацией
  - Интеграция с пользовательским контекстом
  - Статическая типизация
  - Вспомогательная информация доступна в процессе разработки (работает автодополнение и т.д.)
- Недостатки
  - Высокая сложность тестирования
  - Высокая сложность отладки

Цитирование кода (Code quotation)

Итоги

- Есть ли будущее у такого подхода?
  - Какие альтернативы?
  - Нужна ли гомогенность?
  - ...
- Какие потенциальные области применения?
- Не слишком ли сложный механизм для рядового пользователя?

Контакты

- Почта: semen.grigorev@jetbrains.com
- Проект на GitHub:
  https://github.com/YaccConstructor/Brahma.FSharp