Разработка механизма использования OpenCL-кода в программах на F#

Автор: Кирилл Петрович Смиренко, 371 группа
Научный руководитель: к.ф.-м.н., доц. С.В. Григорьев

Санкт-Петербургский государственный университет
Кафедра системного программирования

25 апреля 2017г.

Введение

- Графические процессоры (GPU) – средство ускорения вычислений
- Средства программирования видеопроцессоров: CUDA, OpenCL
- Существуют инструменты запуска высокоуровневого кода на GPU
- Потребность использования низкоуровневого кода в ЯВУ
  - переиспользование кода – хорошая практика
  - переписывать специфические конструкции CUDA и OpenCL на высокоуровневом языке общего назначения нецелесообразно

Обзор: существующие решения

- Средства программирования видеопроцессоров в .NET:
  - Alea GPU (CUDA)
  - Brahma.FSharp (OpenCL)
- Средства запуска CUDA-кода из ЯВУ:
  - CUSP
  - ManagedCuda
- Существующие решения не позволяют типизированно вызывать произвольный низкоуровневый код

Обзор: провайдеры типов в F#

- Генерируют типы данных и встраивает в окружение времени исполнения
- Могут использовать параметры при генерации типов
- Преимущества перед кодогенерацией:
  - интеграция с пользовательским контекстом
  - генерация типов происходит одновременно с компиляцией пользовательского кода
- Недостатки:
  - трудность тестирования
  - высокая сложность отладки

Постановка задачи

Цель: добавление возможности переиспользования OpenCL C-кода в Brahma.FSharp

Задачи:

- Исследовать возможности провайдеров типов F#
- Реализовать лексический и синтаксический анализатор заголовков OpenCL-функций
- Обеспечить возможность типизированного вызова OpenCL-функций в коде на F#
- Провести экспериментальные исследования решения

Лексический и синтаксический анализатор

Инструменты: FsLex, YaccConstructor (язык YARD)
Материалы:

- грамматика C99
- спецификация OpenCL C 2.0

Особенности реализации:

- Синтаксический анализатор описан S-атрибутной грамматикой в EBNF
- Разбираются только заголовки функций

Провайдер типов для OpenCL-функций

- Генерирует F#-функцию, типизированную так же, как исходная функция на OpenCL C
- Провайдер параметризован путём к подгружаемому файлу
- Количество функций в подгружаемом файле не ограничено
- Поддерживается отображение указателей в C в ссылки либо массивы в F#

```
let matvec = KernelProvider<matvecPath, TreatPointersAsArrays=true>.matvec
```

Рис.: Пример использования провайдера типов

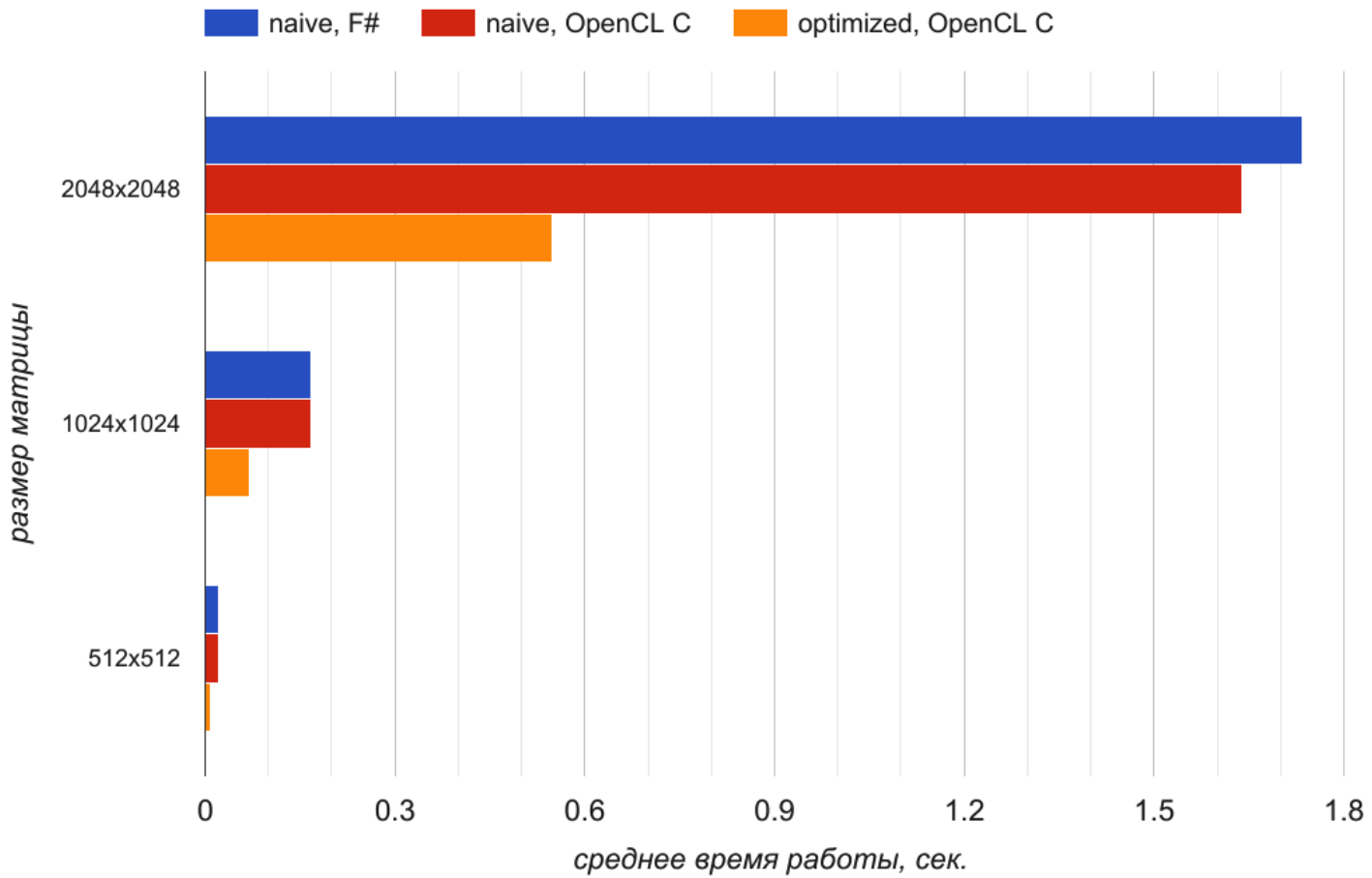
- Ядро Brahma.FSharp доработано для передачи подгружаемых функций драйверу OpenCL

Эксперименты

- Исследование работы реализованного модуля с оптимизированным OpenCL-кодом
- Перемножение вещественных матриц с использованием барьеров и локальных групп OpenCL
- Сравнение с наивными реализациями на F# и OpenCL C

Эксперименты

- NVIDIA GeForce GT 755M, тактовая частота GPU 980 МГц, память 2048 МБ



Результаты

- Исследован механизм провайдеров типов в F#
- Реализован лексический и синтаксический анализатор заголовков OpenCL-функций на языке YARD
- Реализован модуль подгрузки исходного кода на OpenCL C в рамках Brahma.FSharp
- Проведены экспериментальные исследования работы модуля