Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Лунина Полина Сергеевна

Поддержка кортежей и структур в библиотеке Brahma.FSharp

Курсовая работа

Научный руководитель:
к. ф.-м. н., ст. преп. Григорьев С. В.

Санкт-Петербург
2017

Оглавление

Введение

В программировании часто встречаются задачи, позволяющие получить значительный выигрыш в производительности при использовании массово-параллельных архитектур. Одной из самых распространенных является GPGPU — техника использования GPU для общих вычислений, обычно выполняемых CPU. Существуют различные реализации этой техники, основанные на разных языках программирования и предназначенные для выполнения на разных процессорах, например, OpenCL или CUDA. OpenCL (Open Computing Language) [4] — фреймворк для написания программ, связанных с параллельными вычислениями на различных графических и центральных процессорах. OpenCL относится к низкоуровневым языкам, и поэтому он не очень удобен при непосредственном использовании.

Существуют инструменты, позволяющие использовать возможности OpenCL с помощью высокоуровневых языков программирования, например, PyOpenCL, FSCL, JavaCL и другие. Одним из таких инструментов является написанная на языке F# библиотека Brahma.FSharp [1], основанная на транслировании F# quotation в OpenCL.

В настоящий момент в библиотеке Brahma.FSharp реализована поддержка примитивных типов и массивов. Однако в задачах, решаемых при помощи распараллеливания, часто удобно использовать такие типы, как структуры и кортежи, позволяющие представить данные в виде набора переменных различных типов. В Brahma.FSharp не поддерживаются кортежи и существует только частичная реализация поддержки структур.

В данной работе реализована поддержка кортежей и структур и добавлены некоторые функции для их использования.

1. Постановка задач

Цель данной работы — реализация поддержки кортежей и структур в библиотеке Brahma.FSharp. Для достижения цели были выделены следующие задачи:

- изучение библиотеки Brahma.FSharp;
- изучение особенностей языка OpenCL;
- реализация поддержки структур и кортежей в Brahma.FSharp;
- тестирование;
- добавление примеров и описания на официальный сайт.

2. Обзор

Помимо Brahma.FSharp существуют другие библиотеки на F# для интеграции вычислений на графических процессорах: FSCL [2] и Alea GPU [3]. В библиотеке Alea GPU реализована трансляция структур, а в FSCL — структур и кортежей.

В Brahma.FSharp уже реализована трансляция, механизм работы с памятью, а также некоторые функции для обработки примитивных типов данных и массивов. В программах, написанных на F#, также широко применяются кортежи и структуры, и было бы полезно наличие их поддержки и для параллельных вычислений на GPGPU.

Одной из особенностей языка OpenCL является отсутствие такого типа данных, как кортеж, что требует поиска новых решений для реализации их трансляции на FSharp, например, с использованием существующих в OpenCL структур.

3. Основная часть

В данном разделе описаны детали реализации поддержки кортежей и структур в библиотеке Brahma.FSharp.

3.1. Поддержка структур в Brahma.FSharp

Использование структур в библиотеке реализовано следующим образом:

- объявление структуры на F# транслируется в соответствующее объявление на OpenCL;
- при инициализации нового экземпляра структуры на стороне OpenCL создается объект специально созданного типа StructType с доступом к полям структуры, который переводится в структуру, поддерживаемую OpenCL;
- структура загружается в буфер памяти соответствующего размера.

На момент начала работы в библиотеке существовала частичная реализация поддержки структур. Было проведено тестирование и исправлены обнаруженные ошибки при инициализации новой структуры. Таким образом, стало возможным создание структуры с несколькими конструкторами, имеющими разное количество аргументов, например такой:

```
[<Struct>]
type s =
    val x: int
    val y: int
    new (x1, y1) = {x = x1; y = y1}
    new (x1) = {x = x1; y = 0}
```

Инициализировать новый экземпляр такой структуры можно следующими способами:

```
let a = new s(1)
let a = new s(1, 2)
```

Однако сложные конструкторы, использующие создание новой структуры без присвоения ей имени, т.е.

```
let a = (new s(1)).x + 4
```

в данный момент недоступны.

Также был рассмотрен вариант передачи структур на GPU по ссылке, однако это не позволило бы создавать массивы структур, так как в OpenCL нельзя использовать указатели на указатели. Кроме того, изучение спецификации OpenCL 1.2 показало невозможность объявления структуры, полем которой является массив.

Таким образом, в результате данной работы в библиотеке Brahma.FSharp стали доступны: объявление структур, инициализация новых структур вне и внутри kernel-функции, передача в качестве аргументов функции, обращение к полям по имени и их изменение, создание структур с несколькими конструкторами от разного количества аргументов и использование массивов структур.

3.2. Поддержка кортежей в Brahma.FSharp

В языке OpenCL нет такого типа данных, как кортеж, поэтому поддержка кортежей была реализована с помощью нескольких скрытых от пользователя преобразований в структуры следующим образом:

- в отличие от структур, типы элементов кортежа никак не объявляются до непосредственного его появления в программе, поэтому при каждом появлении нового кортежа происходит проверка на существование объявления кортежа с такими же типами элементов и, при отсутствии, он объявляется как структура с именем tuple + уникальный номер;

- в трансляторе создан тип TupleType, представляющий собой надстройку над типом StructType с зафиксированными именами полей (“_1”, “_2” и т. д.);

- для загрузки в память кортеж необходимо перевести в структуру следующим образом: задана специальная структура с полями обобщенного типа (struct t2<T1, T2>), конструктор которой может принимать переменные любого типа; в ходе выполнения определяются типы элементов кортежа, и на их основе создается экземпляр вышеописанной структуры, который и помещается в память.

В данной работе была реализована возможность доступа к элементам с помощью стандартных функций fst, snd и отдельно реализованных функций first, second и third. Важной деталью реализации последних трех функций является то, что хотя их трансляция производится исключительно по имени, необходимо обработать вызов шаблона TupleGet. При этом вызове в связи с особенностью платформы .NET происходит преобразование функции в метод, что приводит к ошибке с несоответствием входных данных, т.е элементы кортежа воспринимаются как несколько отдельных объектов. Данная проблема была решена с использованием сигнатур — специальных конструкций в F#, определяющих типы входных и выходных данных функции.

Таким образом, была реализована поддержка кортежей из двух и трех элементов, а именно: передача их в качестве аргументов функций, возможность доступа к элементам, создание новых кортежей внутри функции и использование массивов кортежей.

4. Эксперименты

В системе NUnit была протестирована основная функциональность реализованных возможностей, а именно:

- объявление структур;
- инициализация структур и создание кортежей вне и внутри kernel-функции и передача в качестве аргументов;
- доступ к полям структур и элементам кортежей;
- объявление массивов кортежей и структур.

Заключение

В рамках данной работы были получены следующие результаты:

- изучены принципы работы библиотеки Brahma.FSharp;
- изучены особенности языка OpenCL;
- реализована поддержка структур и кортежей в Brahma.FSharp;
- проведено тестирование;
- на официальный сайт добавлены примеры использования новых возможностей и описание.

В качестве дальнейшего развития возможны:

- реализация поддержки кортежей из большего количества элементов;
- возможность задания структуры, содержащей массив;
- возможность применения функций fst и snd к кортежу, объявленному в виде let (a,b) = (1,2);
- расширение возможностей использования конструкторов структур;
- реализация механизма передачи массива кортежей с CPU на GPU и обратно.

Список литературы

[1] Проект Brahma.FSharp [Электронный ресурс]. — URL: https://github.com/YaccConstructor/Brahma.FSharp (online; accessed: 10.05.2017).

[2] Проект FSCL [Электронный ресурс]. — URL: https://github.com/FSCL (online; accessed: 10.05.2017).

[3] Сайт проекта AleaGPU [Электронный ресурс]. — URL: http://www.quantalea.com/ (online; accessed: 10.05.2017).

[4] Спецификация OpenCL 1.2 [Электронный ресурс]. — URL: https://www.khronos.org/registry/OpenCL/specs/opencl-1.2.pdf (online; accessed: 10.05.2017).