Санкт-Петербургский государственный университет
Кафедра системного программирования

Brahma.FSharp как основа для обобщённой разреженной линейной алгебры на GPGPU

Панфилёнок Дмитрий Викторович

Научный руководитель: к.ф.-м.н., доцент кафедры информатики С. В. Григорьев
Рецензент: инженер-программист ООО «ИнтеллиДжей Лабс» А. В. Бережных

Санкт-Петербург
2022

Введение

- Использование GPGPU может повысить производительность алгоритмов анализа графов
- Современные инструменты достигают этого ценой более сложной модели программирования

Стандарт GraphBLAS:

- Графы могут быть представлены разреженными матрицами
- Алгоритмы на графах можно выразить в терминах линейной алгебры над различными полукольцами
- Спецификация GraphBLAS описывает структуру объектов и методов для реализации таких алгоритмов в терминах линейной алгебры

Существующие решения

Реализация	Язык	Поддержка GPU
SuiteSparse	C	Нет
GBTL	C++	Нет
GraphBLAST	C++	CUDA
CombBLAS	C++	Частичная, CUDA
Graphulo	Java	Нет

Недостатки существующих решений:

- Отсутствие инструментов, полностью реализующих GraphBLAS на GPU
- Низкая переносимость решений, основанных на CUDA
- Нет самостоятельных библиотек для высокоуровневых языков

Возможное решение

Инструмент: GraphBLAS-sharp
Поддержка GPU: OpenCL
Язык: F#

- Проверки времени компиляции
- Платформа и экосистема .NET
- Поддержка алгебраических типов данных
- Широкие возможности для метапрограммирования

Использует: Brahma.FSharp

Существующие проблемы

1. Отсутствие поддержки произвольных атомарных операций
2. Отсутствие поддержки трансфера пользовательских типов данных из управляемой памяти в видеопамять
3. Неэффективное управление памятью устройства
4. Отсутствие возможности исполнения нескольких OpenCL ядер параллельно

Постановка задачи

Предметом данной работы является улучшение библиотеки Brahma.FSharp с целью её применения в инструментах для обобщённой разреженной линейной алгебры на GPGPU.

Задачи:

- Реализовать поддержку обобщенных атомарных операций
- Реализовать поддержку трансфера пользовательских типов данных
- Обеспечить возможность управления выделением памяти на OpenCL устройстве
- Обеспечить возможность параллельного исполнения OpenCL ядер
- Сравнить производительность полученного решения с аналогами

Особенности:

- Позволяет использовать подмножество языка F# для написания ядер OpenCL
- Позволяет проверять некоторые ошибки на этапе компиляции
- Обеспечивает переносимость за счет кодогенерации

Содержит:

- Транслятор из F# в OpenCL
- API хоста для запуска ядер и управления памятью

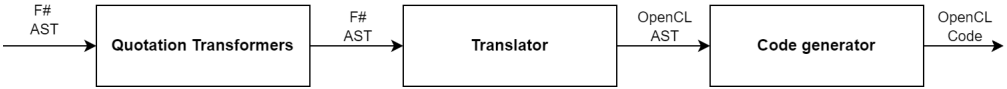

Рис.: Устройство транслятора

Поддержка произвольных атомарных операций

- Стандартные операции → вызов встроенных атомарных функций OpenCL
- Пользовательские операции → вызов сгенерированных функций, реализующих спинлок

```
val atomic : ('a → 'b) → ('a → 'b)
```

Листинг 1: Сигнатура функции atomic

Поддержка трансфера пользовательских типов данных

Методы класса **CustomMarshaler**:

- `IsBlittable(Type): bool` — определяет, является ли тип данных непреобразуемым
- `WriteToUnmanaged('a[]): int` — копирует данные из управляемой памяти в неуправляемую
- `ReadFromUnmanaged(IntPtr, length: int): unit` — копирует данные из неуправляемой памяти в управляемую

Было:

- int
- double
- sbyte
- ...

Добавлено:

- Структуры
- Кортежи
- Записи
- Размеченные объединения

Существующие проблемы, связанные с управлением памятью

- Невозможно выделять память напрямую на OpenCL устройстве
- Невозможно выставлять нужные флаги буфера при создании
- Невозможно автоматически освобождать выделенную память

Управление памятью

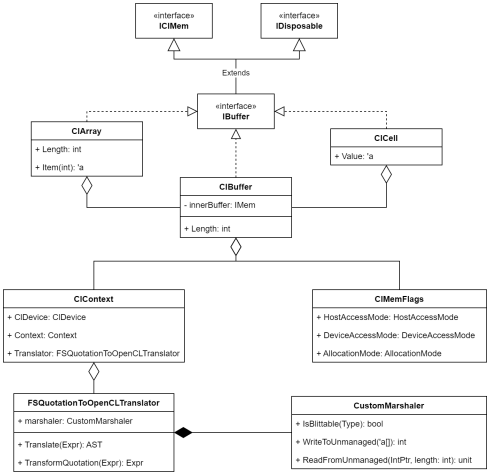

Рис.: Иерархия абстракций для управления памятью

Архитектура решения

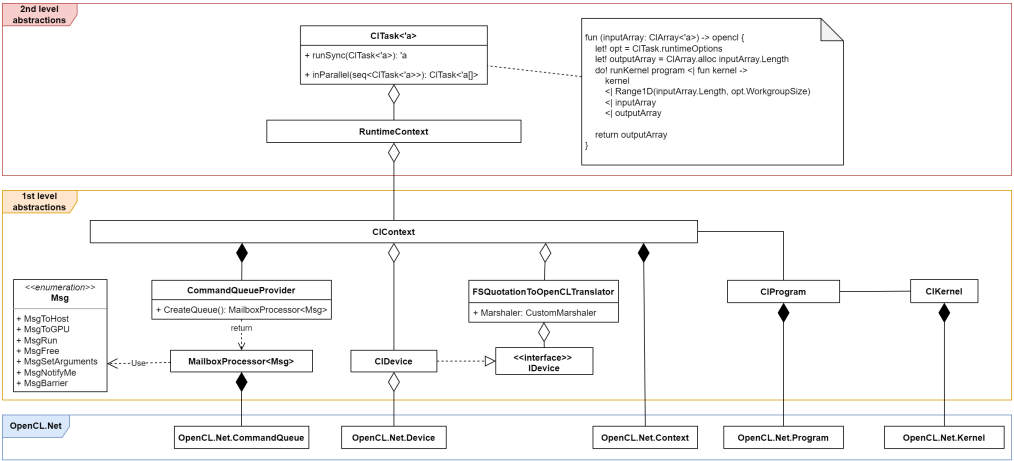

Рис.: Архитектура решения

Сравнение с аналогами

Характеристика	**Brahma.FSharp**	**FSCL**	**ILGPU**
Поддержка непреобразуемых типов данных	Да	Да	Да
Поддержка обобщенных структур данных	Да	*Нет*	Да
Поддержка преобразуемых типов данных	Да	Да	*Нет*
Поддержка произвольных атомарных операций	Да	*Нет*	Да
Поддержка параллельного исполнения команд	Да	Да	Да

Таблица: Сравнение с аналогами

Оценка затрат на трансфер данных: запись

Библиотека	Длина массива	Среднее, мкс	Стандартная ошибка, мкс
Brahma.FSharp	1000	39	3
ILGPU	1000	45	2
Brahma.FSharp	1 000 000	2,507	292
ILGPU	1 000 000	1,082	94

Таблица: Оценка времени записи данных в видеопамять, непреобразуемый тип данных ValueOption<int>

Библиотека	Длина массива	Среднее, мкс	Стандартная ошибка, мкс
Brahma.FSharp	1000	2,685	418
Brahma.FSharp	1 000 000	2,044,113	48,098

Таблица: Оценка времени записи данных в видеопамять, преобразуемый тип данных bool

Оценка затрат на использование атомарных операций

Реализация	**Размер пространства**	**Среднее, мкс**	**Стандартная ошибка, мкс**
Brahma.FSharp (нативная)	1000	138	65
Brahma.FSharp (пользовательская)	1000	96	26
ILGPU (нативная)	1000	20	8
ILGPU (пользовательская)	1000	631	88
Brahma.FSharp (нативная)	100 000	126	40
Brahma.FSharp (пользовательская)	100 000	298	42
ILGPU (нативная)	100 000	14	4
ILGPU (пользовательская)	100 000	541,701	7,757
Brahma.FSharp (нативная)	10 000 000	366	45
Brahma.FSharp (пользовательская)	10 000 000	14,484	4,588

Таблица: Оценка затрат на использование атомарных операций

Результаты

- Реализована поддержка обобщенных атомарных операций
- Реализована поддержка трансфера обобщенных типов данных, таких как структуры, кортежи и размеченные объединения
- Улучшена модель управления памятью
- Переработана архитектура библиотеки
- Реализована возможность параллельного исполнения ядер OpenCL
- Проведено экспериментальное сравнение предложенной реализации с аналогами