Санкт-Петербургский государственный университет
Кафедра системного программирования

Библиотека операций разреженной линейной алгебры с использованием Brahma.FSharp

Кирилл Анатольевич Гарбар, группа 20.Б09-мм

10 июня 2024 г.

Научный руководитель: к.ф.-м.н. С.В. Григорьев, доцент кафедры системного программирования

Рецензент: Е.С. Орачев, инженер-разработчик графики, ООО «ФАРВАТЕР»

Санкт-Петербург
2024

Введение

- **Графы** и **алгоритмы их анализа** применяются во множестве областей дейятельности
- **Матрицы и векторы** — способ параллельной обработки и анализа графов
- Стандарт **GraphBLAS** определяет строительные блоки алгоритмов над графами на языке линейной алгебры
- Множество реализаций, преимущественно на **C/C++**
- **CUDA**, **OpenCL** и графические процессоры
- **GraphBLAS-sharp** — вдохновлённый **GraphBLAS** проект на языке **F#** платформы **.NET**
  - **Brahma.FSharp** — транслятор из **F#** в **OpenCL**

Работа с GPU на .NET

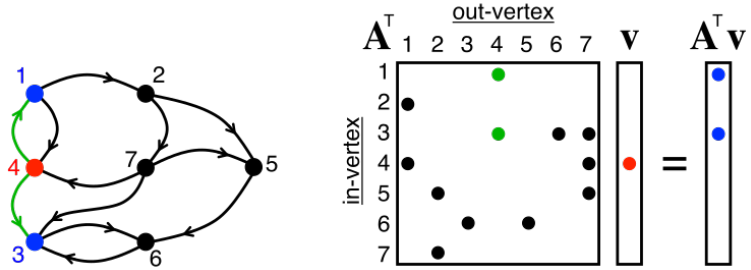
- **OpenCL.NET** — обёртка OpenCL, ядра на языке C
- **GPU.NET/Cudafy** — трансляторы в CUDA, разработка прекращена
- **ILGPU** — активно разрабатываемый транслятор C# в CUDA
- **Brahma.FSharp** — транслятор F# в OpenCL

Библиотеки операций линейной алгебры и анализа графов

Название	CPU	GPU Nvidia	GPU AMD/Intel	Язык
GraphBLAS-sharp	Да	Да	Да	F#
SuiteSparse	Да	Нет	Нет	C
GraphBLAST	Нет	Да	Нет	C++
Spla	Да	Да	Да	C++
CUSP	Нет	Да	Нет	C++
Math.NET Numerics	Да	Нет	Нет	C#
QuikGraph	Да	Нет	Нет	C#

Таблица: Сравнение библиотек операций линейной алгебры и анализа графов по поддержке устройств различных производителей и языку программирования.

Обход в ширину в терминах линейной алгебры



Источник: en.wikipedia.org/wiki/GraphBLAS

Постановка задачи

Цель работы — реализация библиотеки операций линейной алгебры на платформе .NET с использованием OpenCL и её экспериментальное исследование.

Задачи:

1. Доработать существующие операции в GraphBLAS-sharp и реализовать новые
   - Поэлементное сложение разреженных CSR-матриц
   - Умножение разреженной матрицы на плотный и разреженный вектор
2. Реализовать алгоритмы анализа графов
   - Обход графа в ширину
   - Поиск кратчайшего пути из заданной вершины
   - Ранжирование веб-страниц с помощью PageRank
3. Провести исследование производительности библиотеки и сравнить с аналогами

Реализация операций и алгоритмов

Поэлементное сложение CSR-матриц

- Параллельное построчное слияние соответствующих массивов, представляющих матрицы — алгоритм **MergePath**

Умножение на плотный вектор

- Умножение соответствующих значений
- Построчное суммирование значений, полученных на прошлом этапе

Умножение на разреженный вектор

- Отбор строк, участвующих в умножении
  - Оптимизация с применением маски
- Сортировка элементов матрицы по столбцам
- Умножение и суммирование значений

Алгоритмы анализа графов

- Основа каждого алгоритма — умножение матрицы на вектор
- Оптимизация обхода в ширину чередованием плотного и разреженного фронта

Оптимизация Brahma.FSharp и базовых операций

Brahma.FSharp

- Замена асинхронного обработчика сообщений MailboxProcessor на асинхронные каналы .NET
- Предоставлен прямой доступ к очереди команд OpenCL в обход обработчика сообщений

Префиксная сумма и сортировка

- Уменьшено количество запускаемых ядер путём их объединения в более крупные

Постановка эксперимента

- **BenchmarkDotNet**
- **graph-bench**
- **Intel Core i7-4790 CPU**, **3.60GHz**, 4 cores, **8 threads**, **32GB** DDR4 RAM, **GeForce GTX 1070**, 8GB VRAM и **AMD Radeon Vega Frontier Edition**, 16 GB VRAM

Название	Размер	Количество элементов	Заполненность	Средняя степень
wing	62 032	243 088	6,3E-5	3,0
coAuthorsCiteseer	227 320	1 628 238	3,2E-5	7,2
coPapersDBLP	540 486	15 245 729	5,1E-5	56,4
hollywood-2009	1 139 905	113 891 327	8,8E-5	98,9
belgium-osm	1 441 295	3 099 940	1,5E-6	2,1
roadNet-CA	1 971 281	5 533 214	1,4E-6	2,8
road-central	14 081 816	33 866 826	1,7E-7	2,4

Таблица: Графы, на которых производилось сравнение

Результат оптимизации префиксной суммы массива

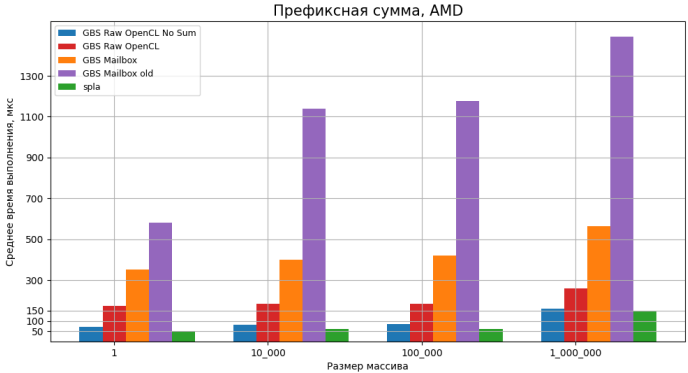


Рис.: GBS Mailbox old — старая версия алгоритма, GBS Mailbox — версия с уменьшенным количеством ядер, GBS Raw OpenCL — алгоритм без использования MailboxProcessor, GBS No Sum — версия без подсчёта суммы элементов.

Результат оптимизации обхода графа в ширину

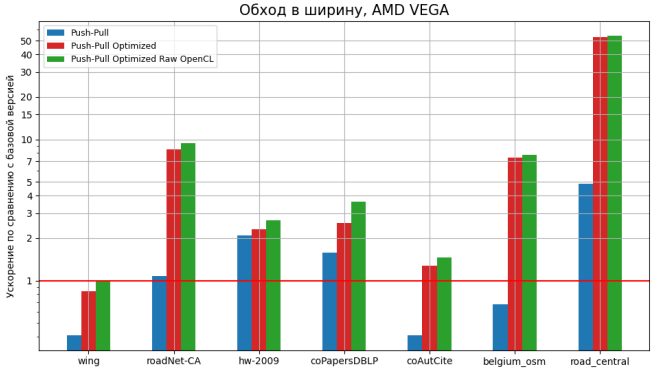


Рис.: Push-Pull — обход с чередованием плотного и разреженного фронта, Push-Pull Optimized — обход с оптимизированной префиксной суммой и сортировкой, Raw OpenCL — версия обхода после отказа от MailboxProcessor.

Сравнение производительности обхода в ширину

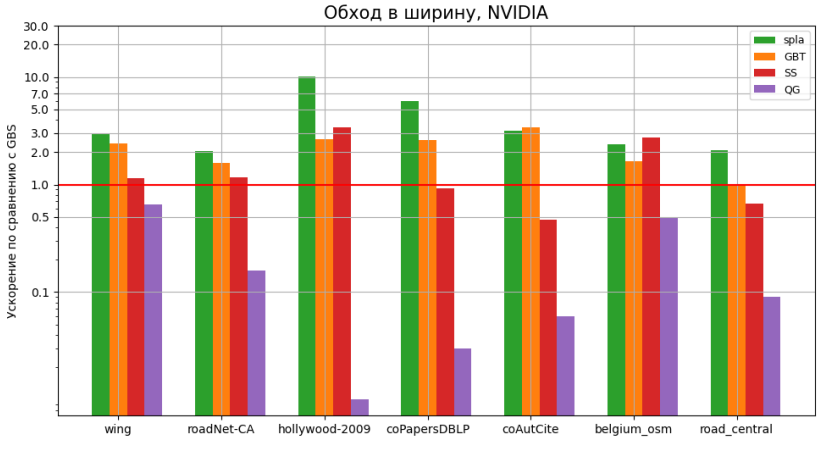


Рис.: Библиотеки: SuiteSpare:GraphBLAS/LAGraph (SS), GraphBLAST (GBT), Spla, QG(QuikGraph).

Сравнение производительности сложения матриц

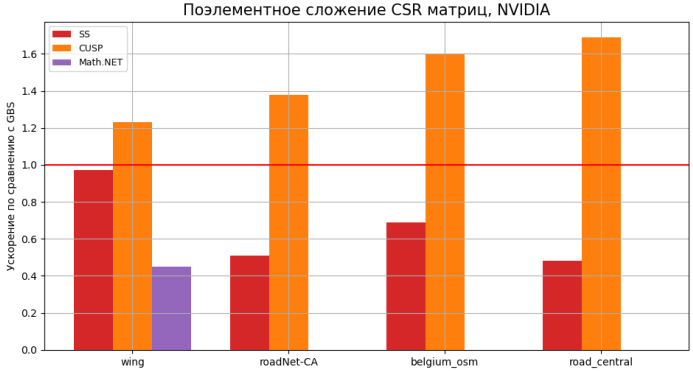


Рис.: Библиотеки: SuiteSpare:GraphBLAS (SS), CUSP, Math.NET Numerics. Отсутствие данных означает, что GraphBLAS-sharp более чем в тысячу раз быстрее.

Результаты

1. Доработаны и оптимизированы имеющиеся операции и реализованы новые
2. Реализованы некоторые алгоритмы анализа графов
3. Проведено исследование и сравнение производительности:
   - Алгоритмы и операции многократно ускорены в результате оптимизаций
   - Производительность библиотеки в среднем в 2-3 раза ниже, чем у аналогов на OpenCL и CUDA, но не хуже, чем у реализаций для CPU
   - Выявлены некоторые причины низкой производительности и способы их устранения

Репозиторий с реализацией:
https://github.com/YaccConstructor/GraphBLAS-sharp/tree/dev
Имя пользователя: **kirillgarbar**