Санкт-Петербургский государственный университет
Программная инженерия

Экспериментальное исследование применимости дистилляции и специализированного аппаратного обеспечения для обработки разреженных данных

Автор: Тюрин Алексей Валерьевич

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

Научный консультант: к.ф-м.н., доцент кафедры СП Д. А. Березун

Рецензент: инженер-программист, ООО «Ланит-Терком» О. В. Медведев

Санкт-Петербург 2022

Разреженная линейная алгебра

- Часто данные содержат много «нулевых» элементов
  - Неэффективно хранить эти элементы в явном виде
  - В случае линейной алгебры данные представлены матрицами
  - Чтобы преодолеть неэффективность хранения таких элементов, матрицы представляются в виде разреженных структур
    - ★ Хранят только нужные элементы, вводя дополнительные указатели
- Подход широко используется
  - Анализ графов
  - Вычислительная биология
  - Машинное обучение
  - ...
- *GraphBLAS* спецификация

GraphBLAS

- Спецификация определяет блоки линейной алгебры для описания алгоритмов анализа графов
  - Использует в реализациях разреженные матрицы
- Оптимизированные блоки $\rightarrow$ оптимизированные алгоритмы
- Предлагает несколько оптимизаций
  - Fusion оптимизации

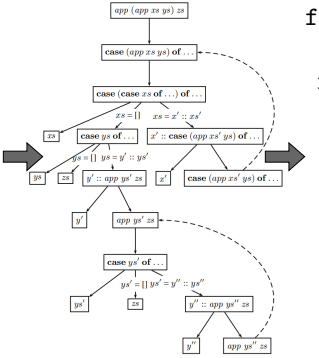
Fusion оптимизации

- Устраняют промежуточные структуры данных
  - Связуют производителей и потребителей
  - `append (append xs ys) zs`
  - Часто возникают в приложениях с разреженной линейной алгеброй
- Ad-hoc реализации
  - Реализованы для определенных операций вручную, например, для применения маски
  - Основаны на правилах переписывания
  - Дублирование производителей, слияние потребителей, слияние результатов производителей (XLA)
  - Автоматический fusion затруднителен из-за адресной арифметики

Дистилляция

```
append (append xs ys) zs where

  append xs ys = case xs of
    []    → ys
    (x:xs') → x : (append xs' ys)
```




```
f xs ys zs where

  f xs ys zs = case xs of
    []    → case ys of
      []    → zs
      y': ys' → y': (g ys') where

        g ys = case ys of
          []    → zs
          y': ys' → y':(g ys')
    x': xs' → x':(f xs' ys zs)
```

- Обобщение суперкомпиляции
- Представляет все возможные трассы исполнения программы конечным графом
- Предоставляет fusion

Специализированное аппаратное обеспечение

- Устройства общего назначения (CPU, GPU) неэффективно используются в разреженных приложениях
  - Малая загруженность
- Специализированное аппаратное обеспечение начинает широко использоваться в разреженных приложениях
- Обращение к памяти остаётся узким местом
- Fusion уменьшает число доступов к памяти

Цель

Цель

Оценить, насколько целесообразно и практично использовать дистилляцию и специализированное аппаратное обеспечение для оптимизации программ с операциями разреженной линейной алгебры

- Изучить подходы к fusion в различных системах
- Реализовать генерацию аппаратной схемы с поддержкой fusion
- Реализовать работу с памятью для схемы
- Реализовать тестовый стенд и провести эксперименты

Реализация генерации аппаратной схемы с поддержкой fusion

- Дистилляция «бесплатно» предоставляет fusion для функциональных языков
- Программа на функциональном языке $\rightarrow$ аппаратная схема?
  - Высокоуровневый синтез
  - Генерация схемы по всей программе повышает отношение «полезных» операций относительно обращений к памяти
  - Общепринятые инструменты для высокоуровнего синтеза императивны

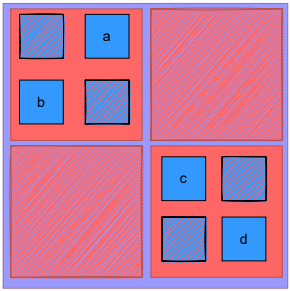
Используемые инструменты

- Дистиллятор[1]
- FHW[2] — компилятор из Haskell в System Verilog
  - Экспериментальный
  - Произвольная рекурсия
  - Параллельное и конвейеризуемое представления потока данных
  - Предполагает наличие аппаратного сборщика мусора (не реализует)
  - Использует External Core GHC в качестве фронтенда
    - GHC > 7.6.3 не поддерживает External Core (сейчас актуальная 9.0.2)
  - Нет поддержки внешней памяти
    - Все данные находятся в коде/схеме

[1] https://github.com/YaccConstructor/Distiller/tree/adding-tests
[2] https://github.com/sedwards-lab/fhw

Структура данных

- Результат fusion зависит от используемого представления разреженных матриц
- Широко используемые координатные форматы (CSR, COO) не подходят
- Используется представление дерева квадрантов (в среднем в 1,5 раза больше памяти, чем координатный формат, на используемых в работе матрицах)



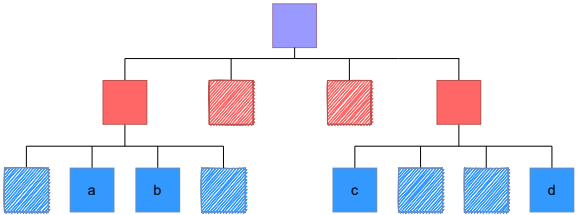
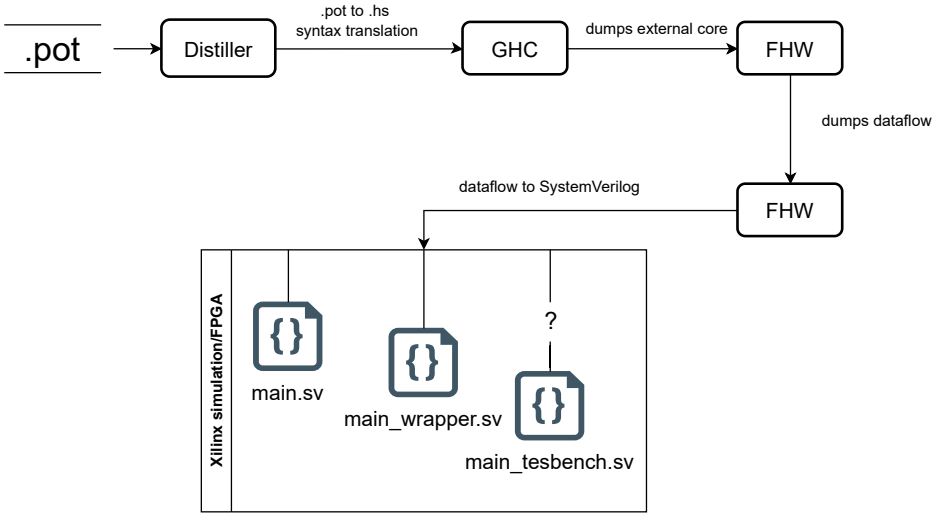


Диаграмма потока данных



Улучшение используемых инструментов

- Дистиллятор
  - Синтаксическая трансляция в Haskell
  - Удаление дублирующихся функций из резидуализированной программы
- FHW
  - Добавлена поддержка алгебраических типов данных с произвольным числом полей
  - Исправлены баги
  - Добавлена возможность передать аргументы в функцию `main`
    - AXI-Stream
    - Для десериализации деревьев с полностью заполненными узлами достаточно одного стека
    - Вся память — bram внутри схемы
- Автоматизирован процесс тестирования и бенчмаркинга
  - Xilinx Vivado + TCL

GHC 7.6.3

- Используется GRIN фреймворк в качестве связующего звена между языком для дистиллятора и представления потока данных
  - Реализован транслятор с языка `.pot` в `.grin`
- Предоставляет дополнительные оптимизации
- CPS и дефункционализация выражаются проще
- В данный момент heap-points-to анализ в GRIN недостаточно производителен для наших примеров[3]

[3] `https://github.com/grin-compiler/grin/issues/128`

Эксперименты

- Даёт ли дистилляция преимущества на программном и аппаратном уровнях?
- Оказываются ли сгенерированные аппаратные схемы производительнее программных реализаций?
- Какие характеристики у сгенерированных аппаратных схем?

Даёт ли дистилляция преимущества на программном и аппаратном уровнях?

Функция	Размеры матриц			Haskell	Haskell (distilled)	FHW		FHW (distilled)	
	m1	m2	m3	время	время	время	время (вкл. IO)	время	время (вкл. IO)
m1 + m2 + m3	64	64	64	29 us	20 us	76 us	170 us	64 us	158 us
	128	128	128	94	79	146	476	134	469
	256	256	256	123	103	202	681	168	662
	512	512	512	219	143	474	1192	375	1093
mask m (m1 + m2)	64	64	64	10 us	7 us	64 us	133 us	46 us	111 us
	128	128	128	38	30	118	322	75	292
	256	256	256	48	42	168	498	104	456
	512	512	512	126	76	400	762	300	729
map f (m1 + m2)	64	64	—	45 us	37 us	189 us	253 us	137 us	202 us
	128	128	—	162	105	524	685	397	579
	256	256	—	312	216	1047	1360	680	986
	512	512	—	436	273	1346	1776	900	1330
map f (kron m1 m2)	2	64	—	64 us	36 us	212 us	242 us	94 us	125 us
	2	128	—	137	68	434	502	199	266
	2	256	—	364	126	1004	1188	449	636
	4	128	—	302	94	694	763	330	401

- До **2**-х и **3**-х раз улучшение времени работы для аппратных и программных реализаций соответственно

Оказываются ли сгенерированные аппаратные схемы производительнее программных реализаций?

Функция	Размеры матриц			Haskell (distilled)	FHW (distilled)		C++
	m1	m2	m3	время	время	время (вкл. IO)	time
m1 + m2 + m3	64	64	64	20 us	64 us	158 us	14 us
	128	128	128	79	134	469	30
	256	256	256	103	168	662	44
	512	512	512	143	375	1093	49
mask m (m1 + m2)	64	64	64	7 us	46 us	111 us	18 us
	128	128	128	30	75	292	33
	256	256	256	42	104	456	46
	512	512	512	76	300	729	65
map f (m1 + m2)	64	64	—	37 us	137 us	202 us	—
	128	128	—	105	397	579	—
	256	256	—	216	680	986	—
	512	512	—	273	900	1330	—
map f (kron m1 m2)	2	64	—	36 us	94 us	125 us	—
	2	128	—	68	199	266	—
	2	256	—	126	449	636	—
	4	128	—	94	330	401	—

- C++ — SuiteSparse
- Сгенерированные аппаратные схемы оказываются хуже в смысле производительности, чем обе программные реализации[4]
- Для `mask m (m1 + m2)` реализация на Haskell с автоматическим fuision близка к реализации на C++

[4] Несколько противоречит `https://dl.acm.org/doi/10.5555/2830840.2830850`

Какие характеристики у сгенерированных аппаратных схем?

Функция	Отношение ($\frac{FHW}{FHW_{distilled}}$)								Частота
	FDRE	LUT3	LUT6	LUT5	LUT4	LUT2	RAMB36E2	MUXF7	
`m1 + m2 + m3`	0,3	0,3	0,3	0,5	0,3	0,3	0,5	0,5	200 МГц
`mask m (m1 + m2)`	0,5	0,5	0,7	0,4	0,7	0,5	0,7	0,6	200 МГц
`map f (m1 + m2)`	1	0,9	0,9	1,2	1	1,1	1,1	1,2	200 МГц
`map f (kron m1 m2)`	1,5	1,5	1,3	2	2	1,8	1,4	1,7	200 МГц

- Процент используемых ресурсов (кроме памяти) $< 1\%$
- Alveo U250 (на других чипах частота ниже, а процент ресурсов выше)

Результаты

- Изучены подходы к fusion в различных системах
- Реализована генерация аппаратных схем с поддержкой fusion[5]
- Реализована работа с памятью для схемы
- Проведено экспериментальное исследование
  - Дистилляция успешно удаляет промежуточные структуры данных для выбранных примеров, сокращая до 2-ух раз число записей/чтений
  - Аппаратные реализации оказываются менее производительными, чем программные, но есть ещё пространство для улучшений в смысле параллелизма и конвейеризации
- Приняты доклады на ICFP 2021 SRC и VPT 2022

[5] `https://github.com/sedwards-lab/fhw`

Даёт ли дистилляция преимущества на программном и аппаратном уровнях?

`m1 + m2 + m3`

Размеры матриц			Отношение ($\frac{FHW}{FHW_{distilled}}$)	
m1	m2	m3	записи	чтения
64	64	64	1,10	1,15
128	128	128	1,02	1,05
256	256	256	1,03	1,06
512	512	512	1,10	1,16

`mask m (m1 + m2)`

Размеры матриц			Отношение ($\frac{FHW}{FHW_{distilled}}$)	
m1	m2	m3	записи	чтения
64	64	64	1,13	1,26
128	128	128	1,06	1,11
256	256	256	1,08	1,09
512	512	512	1,10	1,16

`map f (m1 + m2)`

Размеры матриц			Отношение ($\frac{FHW}{FHW_{distilled}}$)	
m1	m2	m3	записи	чтения
64	64	—	1,10	1,21
128	128	—	1,07	1,14
256	256	—	1,07	1,19
512	512	—	1,10	1,21

`map f (kron m1 m2)`

Размеры матриц			Отношение ($\frac{FHW}{FHW_{distilled}}$)	
m1	m2	m3	записи	чтения
2	64	—	1,71	1,88
2	128	—	1,72	1,87
2	256	—	1,65	1,83
4	128	—	1,81	1,91