ProgrammingTechnologiesOpenCL

examples and other stuff needed for presentation

See documentation at https://madstrix.github.io/ProgrammingTechnologiesOpenCL

Презентация тут: https://prezi.com/view/RhYQD47vnt0MwSNtAHuc/
Ссылки на другие репозитории:

- https://bitbucket.org/kirillovoa/openclexample/overview
- https://github.com/modelflat/OCLRadixSort
- https://github.com/modelflat/coursework2

Speach

slide 0: Overview

Приветствие. Кто нибудь слышал про openCL? Для начала разберемся что такое этот openCL.

slide 1: Что такое openCL?

OpenCL (Open Computing Language) является открытым стандартом для кросс-платформенного параллельного программирования различных процессоров, имеющихся на персональных компьютерах, серверах, мобильных устройствах и встроенных платформах.

Посмотрим красивую **картинку** с сайта openCL. Тут представленны крупные компании-члены консорциума, которые учавствуют в разработке или активно используют данную технологию. А, также, интересные и перспективные разработки, базирующиеся на openCL.

slide 1.1: Цель openCL

OpenCL является открытой стандартизированной технологией для вычислений на специализированных (CPU) и неспециализированных (GPU, APU и др.) устройствах. Это позволяет писать программы для сложных вычислений независимо от используемого оборудования. Открытость данной технологии обеспечивает независимость от разработчика, быстрое исправление ошибок разного рода и поддержку обширного коммьюнити (будь то how-to гайды или обертки для разных языков программирования)

Тут также представлены логотипы компаний, учавствующих в разработке. Если кто-то не успел их рассмотреть. Если столько компаний разрабатывает один продукт, кто же следит за всем этим счастьем?

slide 1.2: Khronos Group

Khronos Group - промышленный консорциум, состоящий из более чем 100 компаний. Целью данной организации является разработка открытых стандартов и интерфейсов программирования в области создания и воспроизведения динамической графики и звука на широком спектре платформ и устройств, с поддержкой аппаратного ускорения. Каждый участник вносит в технологии свой вклад, что обеспечивает хорошую совместимость с платформами разных производителей и позволяет охватывать как можно больше областей применения.

OpenVG предназначен для аппаратно-ускоряемой двухмерной векторной графики мобильных телефонов и смартфонов, медиа- и игровых консолей, и для других электронных устройств.
OpenGL спецификация, определяющая платформонезависимый программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику.
Vulkan кроссплатформенный API для 2D- и 3D-графики, впервые представленный Khronos Group в рамках конференции GDC 2015. Другое название glNext.
OpenVX стандарт компьютерного зрения.
NNEF стандарт для нейронный сетей.

slide 0: Overview

Почему стоит использовать именно openCL, а не другие технологии для вычисления на неспециализированных устройствах?

slide 2: Почему стоит использовать openCL

+ Открытое ПО, разрабатываемое сообществом разработчиков.
+ Хорошая переносимость и высокий уровень абстракции над конкретным оборудованием.
+ Легкая адаптация под различные языки программирования высокого уровня
+ Технология гетерогенного программирования SYCL разрабатывается на базе openCL

Но у всего есть минусы..

slide 2.1: Минусы openCL

- Отсутствие богатой стандартной библиотеки алгоритмов.
- Требует более высокой подготовки программиста для написания оптимального кода.
- Хорошая переносимость достигается за счет небольшого снижения производительности.

slide 2.2: SYCL

SYCL - открытый, кросс-платформенный уровень абстракции, основанный на базовых концепциях, мобильности и эффективности OpenCL, который позволяет писать код для гетерогенных процессоров, используя стандартные функции C++. Гетерогенность - подход, позволяющий выполнять программный код абстрагируясь от специализации процессоров. Таким образом, написанная программа будет выполняться автоматически и на CPU, и на GPU, а работа по менедменту ресурсами ложиться на SYCL.

Ответ на вопрос "почему openCL?" очень прост. На openGL переписывают игры, потому что это более универсльный и мощный инструмент. Такая-же ситуация и с openCL — вполне возможно, что, спустя некоторое количество времени, openCL станет стандартом для крупных компаний, а через еще какой то временной период изчезнет и openCL, сменившись SYCL'ом, попробовать первыми смогут знатоки openCL.

slide 0: Overview

Рассмотрим основные принципы и архитектуру openCL

slide 3: Основные принципы

***Слева** представлена официальная диаграмма кардинальности архитектуры openCL. **Справа** упрощенная диаграмма, достаточная для написания программ базового и среднего уровней, использующих данную технологию.*

На самом деле это все довольно понятно и не так страшно как на этих картинках.

slide 3.1: clPlatform, clDevice & clContext

Для достижения кросс-платформенности, разработчики openCL внедрили уровень абстракции над оборудованием.

- clPlatform - содержит данные производителя (AMD, NVIDIA, Intel) и соединяет вызовы программного кода с соответствующами участками микрокода драйвера производителя.
- clDevice - инкапсулирует конкретное устройство, принадлежащее платформе и содержит еще больше конкретики относительно платформы.
- clContext - абстрактный объект, объединяющий платформу, устройство, kernel и очередь выполнения.

slide 3.2: clProgram & clKernel

*В целом, программа с использованием openCL, состоит из двух (и более) программ. Host - является главной и отвечает за создание контекста и kernel'ей,

подготовку данных и интероперабельность с другими технологиями (например openGL). Kernel - второстепенная программа, выполняющаяся непосредственно на устройстве. Kernel'ей может быть несколько, они могут обмениваться данными между собой. Kernel компилируется непосредственно перед запуском исходя из окружения (clContext), что и обеспечивает кроссплатформенность. Также, программа, написанная на openCL, может полностью использовать память устройств. Для этого существуют модификаторы памяти:

- global - видеопамять, общая для контекста (достаточно медленная);
- local - память виртуального ядра, общая для всех виртуальных ядер (бастрая, меньший объем);
- private - кеш виртуального процессора (очень быстрая, очень маленький объем);
- constant - отдельный логический раздел для констант, располагается или в global, или в local, что определяется производителем.*

slide 0: Overview

`"Хватит болтать - покажи мне код!"`

slide 4: Примеры

openCL имеет огромный функционал, в основном, за счет того, что имеет достаточно мало ограничений. Host программы может быть написан на любом языке программирования, так как любой язык программирования может обращаться к системным библиотекам. Kernel - должен быть написан на языке openCL C 14. Хотя, kernel лишен стандартной библиотеки языка C, но имеет встроенные функции, позволяющие более гибко использовать ресурсы устройства.

slide 4.1: Складывание векторов C

Тривиальная программа складывающая два вектора длинны N, реализованная на языке программирования C.

- Создание `clPlatform` и `clDevice` исходя из аргументов запуска;
- Создание `clContext`;
- Создание `clCommandQueue`;
- Создание `clProgram` и ее компиляция;
- Создание и заполнение массивов данных, перенос данных на устройство;
- Создание `clKernel`;
- Заполнение `clQueue` операциями выполнение kernel'я и считывания данных с устройства;
- Вывод информации о времени выполнения;
- Проверка правильности выполнения;
- Освобождение ресурсов и завершение программы.

А вот как выглядит **вывод** этой программы.

Если кому интересно давайте заглянем в исходные коды: https://madstrix.github.io/ProgrammingTechnologiesOpenCL/d7/d97/trivial_8c_source.html

slide 4.2: Складывание векторов C++

Программа из предидущего примера, но реализованная на языке программирования C++.

- Создание `cl::Platform` и `cl::Device` исходя из аргументов запуска;
- Создание `cl::Context`;
- Создание `cl::CommandQueue`;
- Создание `cl::Program` и ее компиляция;
- Создание и заполнение массивов данных, перенос данных на устройство;
- Создание `cl::Kernel`;
- Заполнение `cl::Queue` операциями выполнение kernel'я и считывания данных с устройства;
- Проверка правильности выполнения.

Если кому интересно давайте заглянем в исходные коды: https://madstrix.github.io/ProgrammingTechnologiesOpenCL/d6/d79/trivial_8cpp_source.html

slide 4.3: Рассчет интегралла Kotlin

- Создание `CLContext` исходя из мощности доступных устройств;
- Создание `CLCommandQueue`;
- Создание и заполнение массивов выходных данных, ассоциация данных с памятью на устройстве;
- Создание `CLKernel`;
- Заполнение `CLQueue` операциями выполнение kernel'я и считывания данных с устройства;

А вот как выглядит **вывод** этой программы.

Если кому интересно давайте заглянем в исходные коды: https://bitbucket.org/kirillovoa/openclexample/src/69cf1b5c537044fbb5186cbbdf5bc563224ce32d/src/main/kotlin/org/bitbucket/kirillovoa/openCLExample/ComputeCL.kt?at=master&fileviewer=file-view-default

slide 4.4: Другие примеры

- Перемножение матриц (C++) : https://github.com/modelflat/ProgrammingTechnologiesOpenCL/blob/master/matrix_multiplication.cpp
- Интероперабельность с openGL :

slide 0: Overview

Ну вот и все. Все, кто хочет более подробно ознакомиться к исходными кодами запысывайте ссылочки)

slide 5: Код тут)

Полные версии исходных текстов и некоторые примеры можно посмотреть по следующим ссылкам:

- https://github.com/modelflat/ProgrammingTechnologiesOpenCL & https://github.com/madstrix/ProgrammingTechnologiesOpenCL
- https://bitbucket.org/kirillovoa/openclexample
- https://github.com/modelflat/coursework2
- https://github.com/modelflat/OCLRadixSort

slide 0: Overview

Спасибо за внимание!