УДК 004.8

**Н.А. Бондарев**

Институт сферы обслуживания и предпринимательства (филиал) ДГТУ

в г. Шахты

*Аннотация. В работе представлен обзор инструментов программирования высокопроизводительных приложений выполняемых на GPU. Представлен краткий сравнительный анализ популярных наборов инструментов разработки приложений для GPGPU.*

*Ключевые слова: высокопроизводительные вычисления, GPGPU, неграфическое программирование на GPU.*

**Введение**

В современном IT-мире видеочипы уже давно заняли обширную нишу в решении вычислительно ёмких задач. Для обеспечения таких вычислений производителями оборудования были разработаны программные инструменты для разработки высокопроизводительных вычислений приложений.

**1 Инструменты неграфического программирования на GPGPU**

**1.1 ATI Stream technology**

**ATI Stream Technology** (ранее как [*ATI*](https://ru.wikipedia.org/wiki/ATI) *FireStream* и *AMD Stream Processor*) — API с открытым исходным кодом, которое предоставляет возможность разработчикам использовать вычислительные возможности GPU . Он предоставляет возможность использования шейдеров графического процессора для запуска вычислительных программ. Интерфейс программирования осуществляется через [OpenCL](https://ru.wikipedia.org/wiki/OpenCL). Это дает возможность ускорения вычислений, и может быть использовано, в том числе, в игровой сфере, для ускорения просчетов физики, если движок физики поддерживает OpenCL.

Областями применения ATI Stream являются также приложения, требовательные к вычислительному ресурсу, такие, как финансовый анализ или обработка сейсмических данных. Использование потокового процессора позволило увеличить скорость некоторых финансовых расчётов в 55 раз по сравнению с решением той же задачи силами только центрального процессора.

**ATI Stream SDK**

ATI Stream SDK v3.0 имеет расширенную поддержку операционных систем, компиляторов, аппаратных компонентов и новейших отраслевых стандартов.

Помимо OpenCL ™ 2.0 в число новых характеристик пакета входят:

* расширенная аппаратная поддержка, включая процессоры архитектуры x86 с потоковыми командами версии SSE2.x и выше;
* дополнительная поддержка компиляторов для Linux и Windows, включая GNU Compiler Collection версии 4.1 или выше, Microsoft Visual Studio (MSVS) и Minimalist GNU (GCC 4.4) for Windows (MinGW);
* поддержка одноканального формата для изображений OpenCL;
* поддержка взаимодействия OpenCL и DirectX 10;
* поддержка дополнительных подпрограмм двойной точности с плавающей запятой в ядрах OpenCL.

**1.2 CUDA**

CUDA ([англ.](https://ru.wikipedia.org/wiki/Английский_язык) *Compute Unified Device Architecture*) — программно-аппаратная архитектура [параллельных вычислений](https://ru.wikipedia.org/wiki/Параллельные_вычисления), которая позволяет существенно увеличить вычислительную производительность благодаря использованию [графических процессоров](https://ru.wikipedia.org/wiki/Графический_процессор) фирмы [Nvidia](https://ru.wikipedia.org/wiki/Nvidia).

CUDA [SDK](https://ru.wikipedia.org/wiki/SDK) позволяет программистам реализовывать на специальном упрощённом диалекте [языка программирования Си](https://ru.wikipedia.org/wiki/Си_(язык_программирования)) алгоритмы, выполнимые на графических процессорах Nvidia, и включать специальные функции в текст программы на Си. Архитектура CUDA даёт разработчику возможность по своему усмотрению организовывать доступ к набору инструкций графического ускорителя и управлять его памятью.

**Программная архитектура**

Первоначальная версия CUDA [SDK](https://ru.wikipedia.org/wiki/SDK) была представлена [15 февраля](https://ru.wikipedia.org/wiki/15_февраля) [2007 года](https://ru.wikipedia.org/wiki/2007_год). В основе [интерфейса программирования приложений](https://ru.wikipedia.org/wiki/Интерфейс_программирования_приложений) CUDA лежит язык С с некоторыми расширениями. Для успешной трансляции кода на этом языке в состав CUDA SDK входит собственный Си-компилятор командной строки **nvcc** компании Nvidia. Компилятор **nvcc** создан на основе открытого компилятора [Open64](https://ru.wikipedia.org/wiki/Open64) и предназначен для трансляции host-кода (главного, управляющего кода) и device-кода (аппаратного кода) (файлов с расширением .cu) в объектные файлы, пригодные в процессе сборки конечной программы или библиотеки в любой среде программирования, например, в [NetBeans](https://ru.wikipedia.org/wiki/NetBeans).

В архитектуре CUDA используется модель памяти [грид](https://ru.wikipedia.org/wiki/Грид), [кластерное моделирование](https://ru.wikipedia.org/wiki/Кластер_(группа_компьютеров)) [потоков](https://ru.wikipedia.org/wiki/Многопоточность) и [SIMD](https://ru.wikipedia.org/wiki/SIMD)-инструкции. Применима не только для высокопроизводительных графических вычислений, но и для различных научных вычислений с использованием видеокарт nVidia. Ученые и исследователи широко используют CUDA в различных областях, включая [астрофизику](https://ru.wikipedia.org/wiki/Астрофизика), [вычислительную биологию](https://ru.wikipedia.org/wiki/Вычислительная_биология) и [химию](https://ru.wikipedia.org/wiki/Вычислительная_химия), [моделирование динамики жидкостей](https://ru.wikipedia.org/wiki/Моделирование_жидкости), [электромагнитных взаимодействий](https://ru.wikipedia.org/wiki/Электромагнитное_взаимодействие), [компьютерную томографию](https://ru.wikipedia.org/wiki/Компьютерная_томография), [сейсмический анализ](https://ru.wikipedia.org/w/index.php?title=Сейсмический_анализ&action=edit&redlink=1) и многое другое. В CUDA имеется возможность подключения к приложениям, использующим[OpenGL](https://ru.wikipedia.org/wiki/OpenGL) и [Direct3D](https://ru.wikipedia.org/wiki/Direct3D). CUDA — [кроссплатформенное программное обеспечение](https://ru.wikipedia.org/wiki/Кроссплатформенное_программное_обеспечение) для таких [операционных систем](https://ru.wikipedia.org/wiki/Операционная_система) как [Linux](https://ru.wikipedia.org/wiki/Linux), [Mac OS X](https://ru.wikipedia.org/wiki/Mac_OS_X) и [Windows](https://ru.wikipedia.org/wiki/Windows).

**Оборудование**

Платформа CUDA впервые появились на рынке с выходом [чипа NVIDIA восьмого поколения](https://ru.wikipedia.org/wiki/GeForce_8) G80 и стала присутствовать во всех последующих сериях графических чипов, которые используются в семействах ускорителей [GeForce](https://ru.wikipedia.org/wiki/GeForce), [Quadro](https://ru.wikipedia.org/wiki/Nvidia_Quadro) и [NVidia Tesla](https://ru.wikipedia.org/wiki/NVidia_Tesla).

Первая серия оборудования, поддерживающая CUDA SDK, G8x, имела 32-битный [векторный процессор](https://ru.wikipedia.org/wiki/Векторный_процессор) [одинарной точности](https://ru.wikipedia.org/wiki/Число_одинарной_точности), использующий CUDA SDK как API (CUDA поддерживает тип double языка Си, однако сейчас его точность понижена до 32-битного с [плавающей запятой](https://ru.wikipedia.org/wiki/Числа_с_плавающей_запятой)). Более поздние процессоры GT200 имеют поддержку 64-битной точности (только для SFU), но производительность значительно хуже, чем для 32-битной точности (из-за того, что SFU всего два на каждый потоковый мультипроцессор, а скалярных процессоров — восемь). Графический процессор организует аппаратную многопоточность, что позволяет задействовать все ресурсы графического процессора. Таким образом, открывается перспектива переложить функции [физического ускорителя](https://ru.wikipedia.org/wiki/Физический_ускоритель) на графический ускоритель (пример реализации — [PhysX](https://ru.wikipedia.org/wiki/PhysX)). Также открываются широкие возможности использования графического оборудования компьютера для выполнения сложных неграфических вычислений: например, в [вычислительной биологии](https://ru.wikipedia.org/wiki/Вычислительная_биология) и в иных отраслях науки.

**Преимущества**

По сравнению с традиционным подходом к организации вычислений общего назначения посредством возможностей графических API, у архитектуры CUDA отмечают следующие преимущества в этой области:

* [Интерфейс программирования приложений](https://ru.wikipedia.org/wiki/Интерфейс_программирования_приложений) CUDA (CUDA API) основан на стандартном языке программирования С с некоторыми ограничениями. По мнению разработчиков, это должно упростить и сгладить процесс изучения архитектуры CUDA
* Разделяемая между потоками память (shared memory) размером в 16 Кб может быть использована под организованный пользователем кэш с более широкой полосой пропускания, чем при выборке из обычных текстур
* Более эффективные транзакции между памятью центрального процессора и видеопамятью
* Полная аппаратная поддержка целочисленных и побитовых операций
* Поддержка компиляции GPU кода средствами открытого [LLVM](https://ru.wikipedia.org/wiki/Low_Level_Virtual_Machine)

**Основные ограничения CUDA:**

* Отсутствие поддержки рекурсии для выполняемых функций;
* Минимальная ширина блока в 32 потока;
* Закрытая архитектура CUDA, принадлежащая Nvidia.

**CUDA Toolkit**

В CUDA Toolkit 8.0 был внесен ряд улучшений расширяющий возможности использования технологии.

**Поддержка архитектуры Pascal**

* Повышена эффективность работы на графических процессорах Pascal
* Упращено программирование использующее общую память, включая поддержку больших объемов данных и одновременного доступа к данным
* Оптимизирована работа API переноса данных из/в общей памяти.
* Увеличена пропускная способность на сверх быстрых скоростях с помощшью NVIDIA® NVLINK ™ , нового высокоскоростного межсоединения.

**Инструменты разработчика**

* Время компиляции NVCC уменьшено до 2x
* Расширена настройка приложений OpenACC и общего кода хоста с использованием новых расширений профилирования

**Библиотеки**

* Ускорение графоаналитических алгоритмов с помощью nvGRAPH

Ускорение приложений Deep Learning с использованием встроеной поддержки FP16 и INT8, поддержка пакетной работы в cuBLAS

**1.3 CUDAfy**

CUDAfy.NET это библиотека позволяющая разрабатывать высокопроизводительные приложения GPGPU полностью из среды Mircosoft.NET.

CUDAfy .NET позволяет легко разрабатывать высокопроизводительныеприложения GPGPU полностью из среды Microsoft .NET. Он разработан на C #.

**Запись кода GPU в .NET для устройств NVIDIA, AMD и Intel**

 CUDAfy позволяет разработчикам .NET легко создавать сложные приложения, которые четко разделяют обработку между хостом и графическим процессором. Для запуска функций устройства GPU не существует отдельных файлов CUDA cu или сложных процедур настройки. Он следует модели программирования CUDA, и любые знания, полученные из учебников или книг по CUDA, могут быть легко перенесены в CUDAfy только в чистом .NET-стиле. CUDAfy поддерживает как генерацию кода CUDA, так и OpenCL и, следовательно, имеет возможность запускать одни и те же приложения на:

* Графические процессоры NVIDIA (CUDA или OpenCL)
* Графические процессоры AMD (OpenCL)
* Процессоры Intel (OpenCL)

**1.4 OpenMP**

OpenMP (Open Multi-Processing) — открытый стандарт для распараллеливания программ на языках C, C++ и Fortran. Дает описание совокупности директив компилятора, библиотечных процедур и переменных окружения, которые предназначены для программирования многопоточных приложений на многопроцессорных системах с общей памятью.

OpenMP реализует параллельные вычисления с помощью многопоточности, в которой «главный» (master) поток создает набор подчиненных (slave) потоков и задача распределяется между ними. Предполагается, что потоки выполняются параллельно на машине с несколькими процессорами (количество процессоров не обязательно должно быть больше или равно количеству потоков).

Задачи, выполняемые потоками параллельно, также как и данные, требуемые для выполнения этих задач, описываются с помощью специальных директив препроцессора соответствующего языка программирования.

**Ключевые элементы OpenMP**

* конструкции для создания потоков (директива parallel),
* конструкции распределения работы между потоками (директивы DO/for и section),
* конструкции для управления работой с данными (выражения shared и private для определения класса памяти переменных),
* конструкции для синхронизации потоков (директивы critical, atomic и barrier),
* процедуры библиотеки поддержки времени выполнения (например, omp\_get\_thread\_num),
* переменные окружения (например, OMP\_NUM\_THREADS).

**1.5 ArrayFire**

ArrayFire – это быстрая библиотека программ для вычислений на графических процессорах (GPU computing) с простым в использовании программным интерфейсом. ArrayFire предоставляет набор функций на базе массива, что существенно упрощает GPU-разработку. Продукт доступен для C, C++ и Fortran, и интегрируется с оборудованием AMD, Intel и NVIDIA.

Библиотека ArrayFire крайне проста в работе. Несколько строк кода в ArrayFire могут заменить десятки строк сырого GPU-кода, что позволит существенно сэкономить время и снизить затраты на разработку.

Преимущества:

* Ускорение процесса написания кода.
* Удобный и простой интерфейс.
* Широкий выбор графических функций.
* Совместимость с любым оборудованием.

Наглядное графическое отображение функций.

**2 Сравнительный анализ инструментов разработки высокопроизводительных приложений GPGPU**

Таблица 1 – Сравнительный анализ инструментов разработки высокопроизводительных приложений GPGPU

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Технология (toolkit) | Поддержка стандартов OpenCl | Язык программирования | Поддержка OS | Поддержка видеочипов |
| ATI Stream sdk 3.0 | + | Brook+ | Windows, Linux | AMD |
| CUDA Toolkit 8.0 | + | Fortran , C, C++, C#(CUDAfy) | Windows, OSX, Linux | Nvidia |
| Array Fire | + | C, C++, Fortran | Windows, OSX, Linux | AMD, Nvidia |

**Заключение**

Обширный инструментарий программирования GPU позволяет решать задачи большой вычислительной ёмкости используя различные среды программирования, операционные системы и оборудование разных производителей.

**Список используемых источников**

1. Технология AMD STREAM URL.: http://www.amd.com/ru-ru/innovations/software-technologies/firepro-graphics/stream (Дата обращения 23.11.2016)
2. APP SDK – A Complete Development Platform URL.: http://developer.amd.com/tools-and-sdks/opencl-zone/amd-accelerated-parallel-processing-app-sdk/ (Дата обращения 03.12.2016)
3. Параллельные вычисления CUDA URL.: http://www.nvidia.ru/object/cuda-parallel-computing-ru.html (Дата обращения 30.10.2016)
4. CUDA URL.:<https://ru.wikipedia.org/wiki/CUDA> (Дата обращения 26.10.2016)
5. ATI Stream: тесты приложений на GPU URL.:<http://www.thg.ru/graphic/ati_stream/ati_stream-01.html> (Дата обращения 21.10.2016).
6. OpenMP URL.:<https://ru.wikipedia.org/wiki/OpenMP> (Дата обращения 03.12.2016)
7. FireArry URL.:http://www.softkey.ru/catalog/programs/142925/arrayfire#.V3 oYc9\_S2b8 (Дата обращения 25.11.2016)
8. NVIDIA cuDNNURL.: <https://developer.nvidia.com/cudnn> (Дата обращения 25.11.2016)
9. NVIDIA ACCELERATED COMPUTING URL.:https://develope r.nvidia.com/cuda-toolkit (Дата обращения 30.10.2016)
10. CUDAfy.NET URL.: https://cudafy.codeplex.com/ (Дата обращения 30.10.2016)