**Инструменты неграфического программирования на GPGPU**

**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.

**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 [Си](https://ru.wikipedia.org/wiki/Си_(язык_программирования)) 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 Си 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)
* Ограничения
* Все функции, выполнимые на устройстве, не поддерживают рекурсии (в версии CUDA Toolkit 3.1 поддерживает указатели и рекурсию) и имеют некоторые другие ограничения

**OpenMP**

OpenMP (Open Multi-Processing) — открытый стандарт для [распараллеливания программ](https://ru.wikipedia.org/wiki/Распараллеливание_программ) на языках [Си](https://ru.wikipedia.org/wiki/С_(язык_программирования)), [Си++](https://ru.wikipedia.org/wiki/Си%2B%2B) и [Фортран](https://ru.wikipedia.org/wiki/Фортран). Дает описание совокупности [директив компилятора](https://ru.wikipedia.org/wiki/Директива_(программирование)), библиотечных [процедур](https://ru.wikipedia.org/wiki/Процедура_(программирование)) и [переменных окружения](https://ru.wikipedia.org/wiki/Переменные_среды), которые предназначены для программирования [многопоточных приложений](https://ru.wikipedia.org/wiki/Многопоточность) на [многопроцессорных системах](https://ru.wikipedia.org/wiki/Симметричная_мультипроцессорность) с [общей памятью](https://ru.wikipedia.org/wiki/Разделяемая_память).

**CuDNN**

Библиотека NVIDIA CUDA® Deep Neural Network (cuDNN) является GPU-ускорением библиотека примитивов для [глубоких нейронных сетей](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=https://developer.nvidia.com/deep-learning&usg=ALkJrhifPGu9cPbtYnQRRY78SCc1yoBi5w). cuDNN обеспечивает очень точную настройку реализации для стандартных процедур, таких как переднюю и заднюю свертке, объединение, нормализации и активации слоев. cuDNN является частью[NVIDIA Deep Learning SDK](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=https://developer.nvidia.com/deep-learning-sdk&usg=ALkJrhjE0JzYAWN0H2LhBgYs0yd0zieU-Q).

Глубокие исследователи обучения и рамочные разработчики по всему миру полагаются на cuDNN для высокопроизводительных GPU ускорения. Это позволяет им сосредоточиться на обучении нейронных сетей и разработке программных приложений, а не тратить время на настройку производительности GPU на низком уровне. cuDNN ускоряет широко используемые глубокие основы обучения, в том числе [Caffe](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=http://caffe.berkeleyvision.org/&usg=ALkJrhhvNJd-vA509Q6gVZoOkYM9OE7T6g), [TensorFlow](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=https://www.tensorflow.org/&usg=ALkJrhi_ONqzMslWTOkC_4qZdruhtBnDHg), [Теано](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=http://deeplearning.net/software/theano/&usg=ALkJrhhW54WeaicFLS8n4bJ2pcMGGEJFRg), [Факел](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=http://torch.ch/&usg=ALkJrhhUM9VB9lSKNCqrGG6nd5t51NlvlQ) и [CNTK](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=http://www.cntk.ai/&usg=ALkJrhi8dug7UPX4lO7juKzgFrWiGpMRtw).

cuDNN находится в свободном доступе для членов [Accelerated программы Developer Computing](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=https://developer.nvidia.com/accelerated-computing-developer&usg=ALkJrhhJFoZaKt4Q3dqobtzOmfSKa5W9sA).

Ученые и исследователи данных могут воспользоваться cuDNN путем загрузки [Deep рамки обучения](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=https://developer.nvidia.com/deep-learning-frameworks&usg=ALkJrhjocRBnQ5vYVomDUxJ_cYkfbS_6CQ) или [NVIDIA Digits](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=https://developer.nvidia.com/digits&usg=ALkJrhjCJ5ZbqLwUh_eR61ytLNkra6lvrQ). Digits позволяет в интерактивном режиме управлять данными, выполнять обучение на нескольких графических процессоров, а также экспортировать наилучшую модель исполнительскую для развертывания без необходимости написания кода.

Ключевая особенность

* Прямые и обратные пути для многих распространенных типов слоев, таких как пулинговой, ЛРН, LCN и пакетной нормализации, Relu, сигмовидной, SoftMax и TANH
* Прямые и обратные свертках процедуры, в том числе кросс-корреляции, предназначенный для сверточных нейронных сетей
* Повторяющиеся Neural Networks (LSTM / ГРУ / РНН), которые обеспечивают до 6-кратного ускорения в Факеле
* Произвольное упорядочение измерение, беговое и субрегионов для 4d тензоров означает простую интеграцию в любой нейронной сети реализации
* Прямые и обратные пути для многих распространенных типов слоев, таких как пулинговой, РЕЛУ, сигмовидной, SoftMax и TANH
* Функции преобразования тензор
* Контекст основе API позволяет легко многопоточности

cuDNN поддерживается на Windows, Linux и MacOS систем с Паскаля, Кеплера, Максвелла, [Tegra K1 или Tegra X1 графических процессоров.](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&prev=search&rurl=translate.google.ru&sl=en&u=https://developer.nvidia.com/embedded/jetpack&usg=ALkJrhgcDuqb_Vu-fNQ37EiwBy2A-rT-Rg)

**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.