<***title***>

[NVidia] CUDA: от API к экосистеме

</***title***>

<***hint***>

Взрывной рост популярности гетерогенных суперкомпьютеров (впрочем как и обычных графических ускорителей) заставил компанию NVidia обновить не только программную модель, но и саму архитектуры ускорителей. В результате графический процессор всё больше напоминает своего центрального собрата, а процесс программирования под оба типа вычислителей уже начинает унифицироваться.

</hint>

<***text***>

Действительно, если первая версия CUDA, представленная 15 февраля 2007 года, являлась скорее ещё одним интерфейсом для программирования видеокарт, то современная CUDA предоставляет доступ к полноценному ускорителю вычислений, который (если нужно) может также задействовать как и видеокарту. За прошедшие четыре года изменения коснулись, в первую очередь, программной части — появилась возможность динамически выделять память, была добавлена частичная поддержка C++, а для особо популярных алгоритмов были разработаны готовые библиотеки. Изменения в аппаратной части хоть и не столь заметны для программиста, но тоже весьма существенны. Так, графический ускоритель обзавёлся кэшами L1/L2, от архитектуры SIMD перешли к более удобной MIMD, появилась поддержка атомарных операций.

Если посмотреть на годовые обороты компании NVidia, то причины подобного внимания к технологии CUDA, являющейся лишь одним из многих направлений, становятся вполне понятными. За каждый год доход от GPGPU-подразделения увеличивается в четыре раза, и при сохранении темпа к 2014 станет вносить основной вклад в бюджет компании. Поэтому нам ещё предстоит услышать явно не один анонс о новых CUDA-решениях от NVidia. А сама идея программирования для видеокарт уже сейчас стала превращаться в полноценную индустрию.

<***subtitle***> Жизнь до CUDA <***/subtitle***>

В последнее время у большинства программистов термин GPGPU (которым любят назвать любую деятельность по использованию графических ускорителей для неграфических вычислений) чётко ассоциируется с решениями от NVidia. Однако данная область начала развиваться ещё за долго до появления CUDA, а первопроходцем в ней была компания ATi, позже ставшая подразделением AMD.

Момент, когда впервые появилась идея задействовать видеокарту не только для визуализации, но и для вычислений вряд ли возможно определить. Толчком послужило появление примерно в 2002-2003 годах новых видеокарт, которые позволяли программно задавать модель визуализации 3D объектов. Изначально для них можно было написать на специальном ассемблере две подпрограммы, которые будут применены к каждому отображаемому треугольнику и к каждому пикселю экрана. Из-за этой узкоспециализированности по началу ограничения были просто ужасными — использовать можно не более 64 вычислительный операций и 32 операций по обращению к памяти, все константы нужно записывать в специальные регистры, а сравнивать любые значения можно только с числом 0.5.

Однако все эти ограничения меркли перед возможностью получить колоссальную по тем временам производительность в 10-20 гигафлопс на одной машине. Поэтому стали появляться новые исследовательские проекты-компиляторы для потоковых языков программирования (разумеется, выполняемых на видеокартах), был запущен авторитетный портал GPGPU.org, а в научно-популярных публикациях всё чаще упоминались возможности по переносу вычислений на видеоускорители. Следующем этапом стало появление шейдерных языков программирования, которые постепенно заменили тот страшный ассемблер и позволили создавать C-подобные программы для видеокарт. Правда, шейдерные языки всё равно предназначались только для визуализации 3D объектов, поэтому идеологам GPGPU всё также приходилось «оборачивать» все свои данные в текстуры и полигоны, а для работы с устройствами использовать DirectX или OpenGL.

Заметив неугасающий интерес к нестандартному использованию видеокарт, компания ATi в 2006 году первой среди производителей «железа» выпустила библиотеку для GPGPU-вычислений. Правда, название было выбрано не очень благозвучным и запоминающимся — DPVM, а поддерживала она, как легко догадаться, только карты от ATi. Но, в любом случае, эта библиотека позволяла задействовать низкоуровневые возможности видеокарт без привязки к 3D графике, оперируя картой именно как ускорителем вычислений. К сожалению, DPVM широкого распространения получить не успела — долгое время каждая страница её документации носила большую и страшную подпись «CONFIDENTIAL», что явно не способствовало росту популярности. В 2007 году её переименовали в CTM (опять-таки одни согласные) и сделали OpenSource-проектом, доступным для всех желающих. Примерно в 2008 последовала резкая смена самой концепции и

дальнейший ребрендинг в уже более знакомый нам Stream SDK. Впрочем, на этом мучения библиотеки не закончились — сейчас она известна как AMD APP SDK.

За это время, пока AMD занималась постоянной переделкой своего инструментария, NVidia выпустила первую версию технологии под названием CUDA, которая оказалась не только более благозвучной, но и гораздо более удобной для программистов. Так и закончилось противостояние проприетарных технологий от NVidia и AMD, фактически даже не успев начаться.

<***subtitle***> Тесла с Ферми <***/subtitle***>

Идейно технология CUDA достаточно проста. В программе с помощью расширений языка C описывается код для выполнения на видеоускорителе. После этого копируются обрабатываемые данные в память GPU и начинается непосредственное выполнение GPGPU-подпрограммы. А в конце забираются уже готовые результаты (ну или по коду ошибки определяется, что же пошло не так). Поддавшись на эту видимую простоту и воодушевившись слайдами из презентаций NVidia (которые обещают 30-кратное ускорение, сотни гигафлопс и прочие блага), многие программисты активно начинают переписывать свои программы с помощью CUDA. И неожиданно узнают об ограничениях, проблемах с совместимостью, а также о том, что на их домашнем компьютере часть функциональности инструментов, которые были так красочно разрекламированы, окажется недоступной. Если к этому добавить ещё и низкую производительность, которая почти всегда присуща «моей первой программе на CUDA», то скептическое отношение многих разработчиков, лично столкнувшихся со всем вышеперечисленным, становится вполне понятным.

На самом деле, не всё так страшно и печально. Видеокарта всё-таки является специализированным ускорителем вычислений, и для достижения хорошей производительности нужно быть готовым разбираться в его архитектурных особенностях и потратить некоторое время на оптимизацию. Во-первых, нужно чётко представлять, что на видеокарте есть 5 типов памяти, и в зависимости от задачи нужно задействовать наиболее подходящие. Во-вторых, нужно понимать, как «порезать» алгоритм на множество легковесных нитей и как эти самые нити сгруппировать в независимые блоки. Ну и в-третьих, следить, чтобы вычисления доминировали над работой с памятью, так как иначе даже самая грамотная оптимизация будет упираться в пропускную способность (которая, стоит отметить, является одной из основных «убийц» производительности). Если программист готов к решению этих проблем, то достижение 10-20% от пиковой производительности становится вполне реалистичной задачей.

Ещё одним моментом, о котором нужно помнить, являются различия в архитектуре видеоускорителей из разных поколений. Хотя CUDA и расшифровывается как “Унифицированная архитектура вычислительных устройств”, у каждого семейства есть свои особенности, в результате чего и появились те самые Тесла и Ферми, в компанию к которым скоро присоединится и Кеплер. Стоит сразу отметить, что сами по себе эти отличия малозаметны. Ну, добавилась улучшенная поддержка операций с двойной точностью, ну появился L1-кэш - программист обо всём этом может вообще и не подозревать. Но это только до тех пор, пока не начнётся процесс оптимизации, там-то все подобные мелочи и дают о себе знать. Наиболее жизненный пример — оптимизация программы, имеющей «прыгающий» паттерн работы с глобальной памятью. Логично предположить, что надо задействовать текстурную память, так как она именно для него и создавалась. И это правильно, но только для «старых» Tesla C1060 — на них действительно удастся получить ускорение в 1.5 или 2 раза. На новых же Tesla C2050/C2070, которые получили известность как Fermi, всё это только замедлит работу, так как на них есть кэш L1, уже превосходно выполняющий функции текстурной памяти. Если копать ещё глубже, то подобных моментов, когда оптимизация под одну архитектуру лишь замедляет работу на других, найдётся достаточно много.

<***subtitle***> Экосистема CUDA <***/subtitle***>

Долгое время CUDA была достаточно узкоспециализированной технологией, знакомство с которой затруднялось не только специфическим диалектом языка C и прочими «заморочками», но и сложностями в настройке и использовании. Например, чтобы добавить поддержку CUDA в среду разработки Microsoft Visual Studio, нужно перекопировать два файла в соответствующие директории, залезть во всевозможные настройки и прописать пути к CUDA SDK, указать правило сборки, выбрать совместимый Runtime, включить подсветку синтаксиса, ну и не забыть «прилинковать» нужные статические библиотеки. У «знающего» программиста всё это займёт около 5-10 минут, у «незнающего» — ровно столько, сколько потребуется для нахождения «знающего» плюс 5-10 минут. Под операционными системами семейства Linux дела обстоят однозначно получше, но всё равно местами требуется «шаманство». Впрочем, поэтому и стали активно использоваться сторонние утилиты типа CMAKE и CudaWizard, легко автоматизирующие весь этот процесс.

Чтобы решить подобные проблемы и сделать технологию CUDA доступной каждому, NVidia пытается создать (и заявляет, что уже создала) так называемую CUDA-экосистему. Что это такое

формально определить сложно, так как само понятие ещё «не устаканилось» и в разных источниках его трактуют по разному. Но сам смысл ясен — сделать среду из взаимодополняющих элементов, полностью решающих основные проблемы CUDA-пользователей. Другими словами, суть новой идеологии сводится к формированию ядра в виде самой технологии CUDA и «наращиванию» поверх него уровней дополнительной функциональности. В такой постановке порог вхождения для новых пользователей существенно снижается — компании NVidia теперь достаточно всего один раз заманить ничего не подозревающих программистов обещаниями стократного ускорения, а как только они столкнутся с какими-либо проблемами, дать ссылку на сайт с описанием соответствующего уровня экосистемы, гарантированно решающего эти проблемы. Причём подобные решения могут быть как от самой NVidia, так и от сторонних разработчиков, распространяться бесплатно или через покупку лицензии и т.д.. Поэтому решившись перенести свой алгоритм на CUDA, вы можете оказаться перед соблазном значительно упростить себе жизнь, приобретя, например, PGI Accelerator за 700$ и ещё CuLa за 99$ в довесок.

В самой NVidia экосистему CUDA разделяют на 4 уровня. Первый содержит расширения для различных языков, всевозможные утилиты, интеграторы в среды программирования и прочие средства упрощения разработки. Здесь как раз и обитает решение озвученной ранее проблемы. Нет желания «в ручную» настраивать Microsoft Visual Studio? Установите NVidia NSight, он не только сделает всё это за вас, но и предоставит возможности по отладке и удалённому запуску. Справедливости ради стоит отметить, что этот плагин вещь действительно полезная, особенно с тех пор, как стал бесплатным. Второй уровень экосистемы включает как готовые проблемно-ориентированные пакеты программ, заточенные под CUDA, так и библиотеки, содержащие CUDA-реализации популярных алгоритмов. Если вам доводилась быть на презентациях от NVidia, то вы обязательно слышали про перенесённые на CUDA пакеты ANSYS/GAMESS, о реализациях алгоритмов FFT, BLAS, Lapack — всё они как раз из этого уровня.

Два последних уровня уже относятся к сферам услуг. Один из них является сетью учебных и исследовательских CUDA-центров, образованных на базе университетов. Всего в мире таких ответвлений более 200, поэтому практически у любого разработчика «по соседству» найдётся подобный CUDA-центр, где он сможет пройти обучение, получить консультацию или просто найти «знающего» программиста. К сожалению, в России есть пока только один официальный центр на базе ННГУ им. Н.И. Лобачевского. Ну и наконец последний уровень экосистемы — коммерческие организации, предоставляющие полный комплекс услуг по CUDA-аутсорсигу. Они готовы и обучить, и проконсультировать, и Теслы поставить, и выполнить за вас все работы переносу ПО. Другими словами, любой ваш каприз будет исполнен, но уже на платной основе.

<***subtitle***> Планы по захвату мира <***/subtitle***>

Резюмируя, хотелось бы ещё раз сформулировать основную идею CUDA-экосистемы — любая проблема может быть легко решена. Если не справляетесь с техническими «заморочками», то установите соответствующие утилиты и инструменты. Если получается низкая производительность, то возьмите готовые реализации алгоритмов. Не хватает опыта — пройдите обучение или получите консультацию. А если всё же не можете самостоятельно перенести на GPU алгоритм, то закажите выполнение этих работ профессиональным разработчикам. И любое из этих действий может быть легко выполнено с помощью различных средств, достаточно лишь зайти на сайт NVidia и выбрать наиболее подходящий вариант.

Насколько данный подход окажется жизненным и востребованным, сейчас сказать сложно, так как и сама GPGPU-индустрия ещё окончательно не сформировалась, да и конкуренты типа Intel и AMD также вынашивают планы по захвату мира, о чём радостно информируют на профильных мероприятиях. Но в любом случае стоит признать, что определённых успехов NVidia всё таки добилась — слово «CUDA» явно стало чем-то большим, чем просто обозначение интерфейса для использования видеокарт.

</text>

<inset>

<***title***> Ответ от Intel <***/title***>

Учитывая рост популярности графических ускорителей, вполне логично, что в закромах у Intel растёт свой «убийца GPU», получивший известность под именем Larrabee. Этот многострадальный проект уже пережил несколько перевоплощений и полное изменение функциональности (и, скорее всего, на этом его зломучения не закончатся). Но он всё равно жив и, если верить анонсам, в начале 2012 будет в каком-то виде официально представлен.

Сам по себе проект уже обрастает легендами. Согласно одной из них, вся эта история началась после передачи архитектуры первого Intel Pentium в Пентагон. Там военные довели его «до ума» - повысили надёжность, снизили энергопотребление и всячески подогнали под свои нужды. Где-

то в середине 2000-ых все наработки из Пентагона перекочевали обратно в Intel, после чего в последней и возникла идея создать свой графический ускоритель. Причём не просто ускоритель, а такой, который обойдёт все аналоги от NVidia и AMD. Было решено построить чип на базе множества x86-совместимых ядер, роль которых отлично выполняла Пентагоновская модификация первого Pentium. Проект получил имя Larrabee и примерно в 2008 году о нём начала появляться информация.

Каждое из ядер, количество которых «бегало» от 8 до 48, умело выполнять за один такт операцию над 512-битные данными (т.е. либо 16 float, либо 8 double). Так как частота была примерно равной 1 гигагерцу, то это обеспечивало пиковую производительность на уровне 2 TFlops. Причём по заявлениям Intel, на тесте SGEMM для матриц размером 4096x4096 было получено 1006 реальных GFlops. Ключевым преимуществом являлась полная совместимость с архитектурой x86 — на этих ускорителях можно запускать обычные программы, написанные с использованием, например, pThreads или OpenMP. Каждый такой ускоритель предполагалась сначала выпускать как плату расширения, а уже потом начать встраивать сразу в материнские платы.

К сожалению, в этом виде проект не дожил до массового выпуска. В конце 2009 года Intel заявила, что архитектуре Larrabee требуется доработка, поэтому сроки его выпуска неоднократно переносились. Летом 2010 года компания Intel ещё раз подтвердила, что проект жив и развивается, но уже под новым кодовым именем — Knight's Corner, и будет предназначаться исключительно для HPC. Примерно полгода назад Intel в очередной раз напомнила о преимуществах своего ускорителя, который после выхода в 2012 году безусловно потеснит графические ускорители. Поэтому нам остаётся только ждать. Ждать, и использовать решения от NVidia, так как им альтернатив пока нет.
</inset>

<inset>
<title> Ответ от AMD ***</title>***

Ещё одним конкурентом для технологии NVidia CUDA и серии Tesla являются, как легко догадаться, решения от AMD. Последняя после покупки в 2006 году канадской компании ATi, занимающейся выпуском видеокарт, получила в своё распоряжение готовую архитектуру графических ускорителей. Причем ускорителей, уже поддерживающих GPGPU и обладающих вполне приличной производительностью. Лидер по тем временам — ATI Radeon X1950 XTX — обеспечивал аж 302 «пиковых» гигафлопса, что всего в 3-4 раза меньше производительности самой современной на сегодняшний день Tesla от NVidia. Впрочем, всё по закону Мура.

Ощутив давление со стороны NVidia, AMD в 2008 году одной из первых объявила о поддержке начавшего тогда маячить стандарта OpenCL. Шаг был вполне логичным — инфраструктура в виде готового Stream SDK уже есть, графические ускорители поддерживают вычисления, почему бы не добавить более удобную программную модель, которая с очень большой вероятностью станет стандартом и потеснит столь ненавистную CUDA? В результате к настоящему моменту у AMD уже есть свой инструментарий, полностью совместимый со спецификацией OpenCL 1.1, и флагманский графический ускоритель Radeon HD 6990, обеспечивающий целых 5100 гигафлопс. Казалось бы, всё отлично, CUDA будет повержена!

Однако при попытке всё это задействовать, на ряде задач (чуть более жизненных, чем перемножение единичных матриц, результат которого игнорируется) неожиданно всплывает столько нюансов, что возникает сильное желание вернуться обратно к NVidia CUDA со всеми её «заморочками», начинающими вдруг казаться столь незначительными и вполне даже логичными. Ещё более страшным оказывается «дутость» гигафлопс, из-за которой пятикратный разрыв в пиковой производительности (5100 против 1003) куда-то резко пропадает — более детальное сравнение можно найти в разделе «Тестовая лаборатория» этого номера.

Поэтому, чтобы окончательно не упустить столь активно развивающийся рынок, AMD выбрала другой путь. Имея в своём составе подразделения по выпуску графических и центральных процессоров, она предложила идею гибридных процессоров, в котором x86-совместимые ядра мирно сосуществуют с графическими на одном кристалле. В результате есть общая память, общие кэши и прочие блоки, а в зависимости от задачи используются нужные ядра — либо сотни графических, либо с десяток обычных. Ну или сразу оба типа вычислителей. Конкурентные преимущества здесь вполне очевидны — не нужно гонять данные по PCI-Express, энергопотребление снижается, процесс программирования упрощается. А главное — у NVidia нет лицензии на выпуск x86-совместимых процессоров, поэтому появления «гибридных Тесел» можно не опасаться.

Стоит заметить, что подобные процессоры, прославившие под рабочим названием AMD Fusion, уже активно выпускаются, правда, пока только для мобильных систем. К примеру, процессор под незамысловатым именем AMD E-350 обладает 2 центральными ядрами (частота 1.6 гигагерц) и 80 графическими (500 мегагерц). Всё это обеспечивает около 85 гигафлопс при энергопотреблении в 18 Ватт. Ровно такую же производительность имеет Intel Xeon X7650, «кушающий» сразу 130 Ватт. После обкатки новой технологии на мобильном сегменте компания AMD планирует плавно переходить к выпуску настольных версий гибридных процессоров. А там уже и до суперкомпьютеров недалеко.

</***inset***>

<***inset***>
<***title***>Камушки от Nvidia</***title***>

После неоднократных попыток со стороны конкурентов создать «убийцу CUDA», NVidia начала периодически покидывать булыжники в соответствующие огороды. Причём если в маркетинговые заявления обычно заворачиваются камушки поменьше, то в антирекламные компании порой вкладываются целые кирпичи. Одним из таких «кирпичей» и стал портал intelsinsides.com, посвящённый шаржам на неудачи Intel, которых, к слову сказать, наберётся не так уж и мало. К сожалению, портал уже год как не обновляется, но за это время там накопилось приличное количество материалов, некоторые из которых приведены в этой статье.
</***inset***>