<***title***>Битва OpenCL'ей | Зелёно-синя-красная OpenCL</***title***>

<***hint***>
Чем популярнее становится стандарт параллельного программирования OpenCL, тем больше производителей «железа» начинают его поддерживать. А это в свою очередь повышает его популярность — ну и так по кругу. В этом году компания Intel выпустила свою первую версию OpenCL SDK, в результате чего у разработчиков появилась возможность выбора — инструментарий от AMD, NVidia или же от Intel?
</***hint***>

<***text***>
Основным и неоспоримым преимуществом OpenCL является всеядность. Вы один раз напишете свою программу, а после этого можете её запускать хоть на многопроцессорных системах, хоть на графических ускорителях. А в будущем — даже на целых кластерах, и такие версии OpenCL разрабатываются. Однако для всего этого потребуется драйвер от производителя «железа», который автоматически отобразит вашу программу на используемый вычислитель.

В настоящий момент подобные OpenCL-драйвера уже выпустили NVidia, AMD и Intel. Реализация OpenCL от NVidia является тонкой прослойкой над их собственной технологией CUDA. Правда, тонкой относительно — NVidia OpenCL как-то уживается в одном SDK с CUDA. А вот при вычислениях прослойка вдруг оказывается очень даже толстой — переход от родной технологии к общепринятому стандарту обычно оборачивается 2-кратной потерей производительности. Преимуществом версии OpenCL от AMD, как легко догадаться, является поддержка и графических ускорителей, и центральных процессоров. Что вполне логично, так как компания не только разрабатывает оба типа этих вычислителей, но и активно пытается их «поженить», превратив в гибридный APU. В отличие от своих конкурентов, Intel как всегда делает ставку на скорость — её OpenCL хоть и поддерживает только CPU, но, по словам представителей компании, делает это превосходно.

<***title***>Тестовые системы</***title***>

Итак, мы решили сравнить все эти три реализации OpenCL. А именно взять набор тестов и без перекомпиляции прогнать их на всех доступных вычислительных устройствах с использованием всех OpenCL-драйверов. Правда мировой интриги как таковой здесь нет — заранее можно предугадать, что графические ускорители обойдут центральные процессоры, а при вычислениях с двойной точностью производительность GPU будет сильно проседать. Однако это предоставляет возможность сравнить в абсолютно равных условиях как CPU-версии OpenCL от Intel и AMD, так и графические ускорители от NVidia и AMD. А также проверить работоспособность и совместимость всех версий OpenCL.

Для тестов нами была собрана система на базе процессора Intel Xeon E3, являющегося одним из первых серверных процессоров с поддержкой расширений AVX (Advanced Vector eXtensions). Данные расширения пришли на смену SSE, увеличив размерность векторов с 128 до 256 бит. Другими словами, теперь за один такт ядро процессора может обрабатывать сразу 8 чисел с одинарной точностью или 4 числа с двойной. И если раньше для использования подобных расширений нужно было переписать код в ассемблерном стиле, то теперь эта обязанность плавно переезжает в OpenCL-драйвер. Вы определяете гранулярность параллелизма на высокоуровневом языке, а уж драйвер пусть сам всё переписывает. Правда, как показали наши тесты, на данный момент он это делает не очень хорошо. А точнее — ужасно.

В качестве графических ускорителей были взяты NVidia GeForce 580GTX и AMD Radeon HD5850. Получилось не совсем честно, так как топовый вычислитель от NVidia сравнивается с чуть-чуть устаревшим конкурентом. Проблема заключалась в чересчур агрессивной маркетинговой политики NVidia, в результате которой самые новые и мощные карты от AMD не столь распространены в научном сообществе, как, например, последние модели ускорителей NVidia. С другой стороны, теоретические производительности этих двух вычислителей весьма и весьма схожи, поэтому объективность не страдает. Почти не страдает.

Для тестов был выбран уже использованный нами ранее бенчмарк SHOC (Scalable HeterOgeneous Computing), который продолжает развиваться при поддержке небезызвестной лаборатории ORNL. Он содержит набор независимых тестов, целью которых является как оценка реально достижимой производительности, так и скорости при решении обычных задач типа перемножения матриц, вычисления быстрого преобразования Фурье и выполнении свёртки больших массивов.

<***title***>Время компиляции и накладные расходы</***title***>

Первый момент, на который хотелось бы обратить внимание — это компиляция вычислительных ядер. В соответствии с парадигмой OpenCL, весь вычислительный код должен динамически компилироваться при запуске программы под целевой вычислитель. Собственно, это и есть одна из основных задач OpenCL-драйверов — зная специфику «железа», заоптимизировать под него программу. Минусом данного подхода являются жёсткие временные ограничения, так как компиляция должна быть очень и очень быстрой. В противном случае вполне возможна ситуация, когда ради 1 секунды вычислений нужно ждать по 5-10 секунд, пока программа оптимизируется под вычислитель.

По данному критерию безусловным лидером оказалась NVidia – время компиляции большого тестового ядра составило порядка 0.1 миллисекунды. Чуть более плохой результат показала версия OpenCL от AMD, которой потребовалось от 0.3 до до 0.9 миллисекунд. Реализация от Intel, напротив, компиляции уделила целых 3.2 миллисекунды. И хотя это в почти в 10 раз больше времени AMD, последующий 2-3 кратный отрыв по производительности на всех тестах наводит на мысль, что эти миллисекунды были потрачены явно не зря.

Другим достаточно важным показателем является время накладных расходов при запуске вычислительного ядра. Понятно, что если вычисления длятся более секунды, то задержки особо и не страшны. Однако иногда встречаются алгоритмы, где требуется частая синхронизация, и как следствие, время работы вычислительных ядер измеряется миллисекундами. Здесь-то накладные расходы и становятся заметными.

Как показали тесты, про подобные проблемы можно смело забыть при использовании OpenCL от Intel и NVidia, а также CPU-версии OpenCL от AMD, где накладные расходы на запуск ядра составили менее микросекунды. Однако при использовании графического ускорителя Radeon 5850HD задержка по непонятным причинам возросла до 1.2 миллисекунд. Если переформулировать — на каждые 800 запусков ядра тратится порядка 1 секунды, что в некоторых случаях достаточно неприятно. К примеру, в области обработки видео-данных можно найти много алгоритмов, требующих запуск десятков тысяч ядер, суммарное время выполнения которых не превышает 10 секунд. Если подобные алгоритмы портировать на графический ускоритель от AMD, то более 50% времени программа будет простаивать, ожидая ответа от OpenCL. Правда, только до тех пор, пока AMD'шники не выпустят исправленную версию драйвера.

<***title***>Зелёный против красного</***title***>

Как уже было отмечено выше, сравнивать напрямую производительность CPU и GPU не совсем этично, поэтому сначала проанализируем результаты работы бенчмарка на графических ускорителях. Итак, GeForce 580GTX - флагман от NVidia, который с лёгкостью обойдёт любую карту серии Tesla, обладает 1.6 терафлопсами пиковой производительности при вычислениях с одинарной точностью. Правда, как и во всех GeForce'ax, при переходе к двойной точности производительность падает ровно в 8 раз — почти до 200 гигафлопс. Его оппонентом стал ускоритель от AMD Radeon 5850HD, который благодаря иной архитектуре достигает намного больших значений пиковой производительности — 2 терафлопса и 400 гигафлопс при использовании типов float и double соответственно. Тест MaxFlops, целью которого является подбор очень простого вычислительного ядра, обеспечивающего максимальную производительность, показал, что заявленные характеристики у обоих вычислителей вполне достижимы. Более того, в большинстве случаев эффективность оказалась порядка 98%. Linpack'у остаётся только завидовать.

А вот при переходе к алгоритмам из жизни, как и следовало ожидать, наблюдается резкий провал в производительности. Впрочем, провал как у NVidia, так и у AMD — всё-таки архитектура GPU идеально подходит только к очень узкому классу задач, алгоритмы из реальной жизни в который попадают крайне редко. Итак, при перемножении матриц в аналогии с функциями SGEMM/DGEMM из пакета BLAS удалось «выжать» всего несколько сотен гигафлопс, причём NVidia в 3 раза обошла решение от AMD. В тесте с вычислением быстрого преобразования Фурье дела оказались ещё хуже — производительность не превысила 70 гигафлопс. Правда, это всё равно позволило видеокартам обойти центральный процессор в 15-30 раз.

В другой группе тестов, где акцент делается не на вычислительную мощь, а на эффективную работу с памятью, у ускорителя от AMD начались непонятные проблемы. К примеру, при решении задачи N тел (где N равно 16000) Radeon 5850HD оказался в 11.5 раз медленнее аналога от NVidia, а что более страшно — он даже не сумел обойти центральный процессор. Более того, при свёртке больших массивов GPU от AMD оказывается уже в 20 раз медленнее CPU от Intel. Складывается впечатление, что подобные проблемы вызваны отнюдь не аппаратной, а программной частью. В данном случае OpenCL-драйвер не смог грамотно отобразить вычислительное ядро, что и вызвало столь резкий провал производительности. Эта утверждение частично подтверждается теми же тестами, но проведёнными с использованием двойной точности, где сразу наблюдается привычная картина — AMD всего в 3-4 раза медленней NVidia, но зато в 6 раз обходит Intel. Примерно это мы и видели в первой группе тестов, измеряющих чистую производительность.

Отдельно хотелось бы остановиться на пропускной способности различных типов памяти. Стоит напомнить, что GPU обладает тремя типами памяти — внешней памятью типа GDDR5 (так называемая глобальная), встроенной в чип разделяемой памятью (локальная, выполняющая роль программируемого кэша) и неким гибридом под названием текстурная память, работающим как глобальная, но имеющим встроенный механизм кэширования. Тесты показали, что наивысшую пропускную способность на всех GPU имеет глобальная память, что несколько странно — уж кто-кто, а по этому показателю явно должен лидировать кэш, а именно локальная память. И проблема опять вызвана программной составляющей — при смене API с OpenCL на более родную CUDA локальная память на GeForce'е действительно начинает лидировать, достигая 490 GB/s вместо полученных ранее 100 GB/s. Другими словами, одна ошибка OpenCL-драйвера, и производительность падает в 5 раз. Всё-таки статическая компиляция имеет свои плюсы.

<***title***>И красный против синего</***title***>

А теперь осталось самое интересное — один центральный процессор, одна тестовая программа и две реализации OpenCL. Условия чуть более чем равные, поэтому можно смело говорить, что победитель является самой быстрой версией OpenCL для центральных процессоров. И победителем становится Intel (пам, па-па-пам). Их реализация оказалась более быстрой во всех тестах, за исключением теста на пропускную способность памяти. Стоит отметить, что в большинстве случаев отрыв составляет от 2 до 3 раз, поэтому если вы используете OpenCL от AMD для вычислений на центральных процессорах, то замена всего одной константы (точнее, смена версии OpenCL) многократно ускорит ваши программы.

Из минусов стоит отметить отсутствие полноценной поддержки AVX — обещанные 100 гигафлопс так и не были достигнуты. Хотя при вычислениях с двойной точностью дела обстоят получше — полученные на тесте MaxFlops 50 гигафлопс означают, что за такт всё-таки удалось обрабатывалось ровно по 4 double-числа. Другим интересным моментом оказалась бесполезность текстурной памяти — на процессоре у глобальной памяти уже есть кэш (и даже не один), и попытка ещё более заоптимизировать обращения к RAM лишь замедлила пропускную способность. К сожалению, данный бенчмарк не измеряет латентность, поэтому оценить возможную область применения для данного класса памяти не удастся.

***

Проведённый тесты показали, что OpenCL работает. Правда, работает не всегда стабильно и медленно — достаточно часто переход на конкурирующую технологию типа NVidia CUDA или Intel C++ позволяет не только упростить жизнь программиста, но и получить дополнительное, иногда N-кратное ускорение. И это вполне логично, хоть какая-нибудь плата за универсальность да должна быть.

Если верить анонсам, то количество разновидностей OpenCL в ближайшем будущем будет активно расти. Так IBM делает свою версию под PowerPC, активно развиваются научные проекты по созданию экзотических реализаций OpenCL, при поддержке Nokia и Samsung начинает набирать популярнось WebCL. Также практически нет сомнений, что производительность и качество оптимизации протестированных нами версий OpenCL от AMD, NVidia и Intel будет только расти, и вполне возможно, что через год или два отличий между стандартом и родными для «железа» технологиями почти не останется. Зато останется совместимость между различными платформами и возможность запуска программы на любом устройстве, будь то петафлопсный суперкомпьютер или мобильный телефон. И опять же — без перекомпиляции.
</***text***>

<***inset***>
<***title***>Выбор SDK</***title***>

Все версии OpenCL совместимы между собой благодаря технологии ICD – Installable Client Driver. Какой бы SDK при разработке своей программы вы ни использовали, она всегда сможет «переключиться» на другую реализацию OpenCL, вызвав соответствующие функции из API. Как результат, вы можете разработать и скомпилировать программу с помощью SDK от AMD, после чего запустить этот же бинарный файл на системе с NVidia Tesla. И всё будет работать.

Поэтому вполне может возникнуть вопрос — какой SDK выбрать? Ответ очевиден для случая, если вы разрабатываете программу под конкретное «железо», так как остаётся только один вариант — использовать SDK от того же производителя. Если же подобной привязки нет, то стоит оценить свои потребности. В SDK от Intel пока очень мало примеров и внятной документации, зато делается акцент на вспомогательные инструменты, которые помогут вам провести ручную оптимизацию вычислительных ядер. В AMD APP SDK наоборот, очень много примеров (причём достаточно полезных) и присутствуют полные и понятные руководства, описывающих как основы, так и тонкости

программирования с использованием OpenCL. Версия же OpenCL от NVidia поставляется вместе с CUDA, поэтому наследует от неё не только примеры, но и документацию и утилиты (правда, «причёсанные» соответствующим образом), что делает данный вариант оптимальным для программистов, уже привыкших к стилю NVidia GPU Computing Toolkit.
</***inset***>