<***Authors***>
Максим Кривов, Андрей Казеннов
</***Authors***>
<***Title***>
Портируем на GPU и оптимизируем для CPU | Портируем и оптимизируем
</***Title***>
<***Hint***>
Достаточно часто слышны заявления от NVidia, что вычислительное будущее за GPU. Intel акцентирует внимание, что основным вычислителем является всё-таки процессор, и без хорошего CPU нынче никуда. Летом появился исследовательский отчёт IBM, что при должной оптимизации CPU сопоставим с GPU по float-производительности. А Power7, само собой, обходит и Xeon, и GeForce. Это мы и решили проверить.
</***Hint***>
<***Text***>
Проверяться, конечно, будут не маркетинговые заявления (что слон сильнее кита, особенно на Марсе в зимнее время года), а тезис о соответствии производительности CPU и GPU при должной оптимизации. Не секрет, что современный процессор может выполнять десятки инструкций за такт, а в одну SMP-систему умудряются засунуть до 8 CPU. Если перемножить, то получается около 100 «ядер», которые вполне соответствуют 200-500 CUDA-ядрам графического ускорителя.

Чтобы тестирование было честным, было решено взять идеальное для GPU приложение (более того, заоптимизированное самими сотрудниками NVidia), и переписать его под CPU. Таким приложением оказался пример NBody из NVidia CUDA SDK, который показывает колоссальную производительность и хоть как-то соотносится с реальными задачами. Заоптимизирован он превосходно — из алгоритма на 20-30 строчек получился 500-строчный *cu*-файл, а достигнутым результатам посвятили целые 20 страниц в книге «GPU Gems 3».

<***SubTitle***> CPU-оптимизация — это не страшно </***SubTitle***>

Прежде чем переходить к непосредственному сравнению китов и слонов (результат которого и так очевиден, особенно если посмотреть на график справа<←***поправить***>), хотелось бы сказать пару слов о проведённой оптимизации. Последовательный код, который шёл вместе с примером NBody, был подвергнут ряду модификаций. Структуры данных были переупорядочены, циклы верхнего уровня распараллелены с помощью OpenMP, нижнего — векторизованы через SSE-вставки, а штатные компиляторы (Visual C++ и gcc) были заменены на более быстрый Intel C++. На всё-про-всё ушло не более 3 часов, плюс ещё один час на финальный тюнинг профайлером. И результат получился впечатляющим — ускорение по сравнению с последовательным кодом составило от 66 до 210 раз. И это без MPI, всё на одном узле!

Если таким же образом посчитать ускорение на картах NVidia, то оно составит примерно 400-600 раз. Победа GPU налицо. Однако, если посмотреть на презентации NVidia, то хорошим результатом от перехода на GPU признаётся даже ускорение в 30-100 раз. А после идёт ма-а-аленькая звёздочка с пояснением, что в CPU-версии использовалось только одно ядро процессора. Подобные звёздочки появляются явно не от хорошей жизни, и связаны они, в первую очередь, с большими трудозатратами на качественную GPU-оптимизацию. Иногда дешевле и проще остановиться на на 100-кратном ускорении, чем потратить дополнительные N дней и «выжать» ещё пару-тройку сотен гигафлопс, доведя итоговое ускорение до красивого значения.

После подобных слайдов сразу же появляется желание построить что-то типа графика «зависимость ускорения от потраченного времени программиста». И начинают мучить смутные подозрения, что на начальных этапах этого графика лидером будет явно не NVidia. Поэтому, если вы получили доступ на гетерогенный кластер, то перед переносом всех своих программ на CUDA стоит задаться вопросом — а оно вам надо? Вполне может оказаться, что потратив явно не один день на портирование, вы получите такое же ускорение, как и после пары часов оптимизации кода под новые центральные процессоры. Понятно, что если вам нужны сотни терафлопс, то без графических ускорителей никуда. Но если вы просто хотите уменьшить время работы существующей программы в 5/10/15 раз, то CPU-оптимизация может оказаться разумным выбором.

<***SubTitle***> CPU против GPU </***SubTitle***>

Гипотеза не подтвердилась — GPU обходит CPU. Отрыв, правда, не такой уж и большой — лидер среди CPU-систем, машина от AMD с 8 процессорами, всего в два раза медленней, чем старая Tesla C1060. Если сравнивать с более новой Tesla C2050, то отрыв увеличивается до 3 раз. В принципе, результат вполне ожидаемый — пиковые производительности соотносятся как 1:3, а эффективность использования всех систем колеблется около 50%. Если брать менее экзотические инсталляции, в которых всего 2-4 процессорных гнезда, то преимущество GPU над CPU будет

примерно 3-6 кратным. Это уже более жизненная ситуация, так как каждом узле гетерогенного кластера обычно есть пара мощных центральных процессоров и 2-3 ускорителя. Поэтому если вам удастся качественно портировать программу на GPU, то следует ожидать ускорение примерно раз в 10. Если получилась более внушительная цифра (25/50/100), то, безусловно, это приятный результат, о котором стоит рассказать на конференции, но вызван он неэффективным использованием центрального процессора.

Может показаться, что подобные сравнения не совсем корректны — взяли кучу процессоров и сравнили с одним единственным ускорителем. А вот если взять ещё парочку ускорителей, то видеокарты «порвут» центральные процессоры как Тузик грелку. Это верно, но в данном обзоре системы рассматриваются с точки зрения программиста. Сколько бы ни было центральных процессоров, это всё одна SMP-система, где каждый поток может адресовать любую ячейку памяти. Понятно, что время доступа будет разное, кэши разделены, и подобную систему формально стоит отнести к классу ccNUMA. Но для программиста это SMP. В случае же нескольких графических ускорителей, на плечи программиста ложатся операции по копированию памяти, так один ускоритель не может адресовать память другого. И в результате не только алгоритм придётся менять, но и сама программа превратиться во что-то большое и MPI-подобное. Поэтому сравнение систем в рамках одной программной модели если и не совсем корректно, то вполне жизненно.

<***SubTitle***> AMD против Intel </***SubTitle***>

Так как были протестированы 4 CPU-системы, то сложно отказаться от удовольствия и сравнить решения двух вечных конкурентов. Правда, сравнивать приходится дорогие и быстрые процессоры с дешёвыми и медленными. Если посмотреть на график, то легко заметить, что процессорам от AMD требуется в два раза больше ядер, чтобы догнать собратьев от Intel. И это вполне логично, так как сильными сторонами AMD является низкое удельное энергопотребление и высокая плотность компоновки в стойках. Это ценно в действительно больших кластерах, но при попытке сравнивать производительность отдельно взятого узла безусловным лидером будет Intel. Их процессоры и большее количество инструкций за такт выполняют, и частота у них повыше. Также они поддерживают технологии HyperThreading и TurboBoost, что позволяет ускорить даже не очень хорошо оптимизированные программы. Результат всего этого — 8 процессоров от AMD с трудом обходят 4 процессора от Intel, имеющих такое же количество ядер и такую же частоту. С другой стороны, 8-процессорные системы от AMD редкостью не назовёшь, в то время как 8 процессорных гнёзд в сервере Intel – это скорее из области фантастики.

Также хотелось бы остановиться на масштабируемости теста при таком большом количестве ядер. Почти на всех системам было получено линейное ускорение, а на 12-ядерном сервере от AMD даже суперлинейное (в 15 раз на 12 ядрах). При этом у двух систем наблюдались периодические «провалы» производительности. Так, на Intel'овском сервере технология HyperThreading лишь замедлила выполнение — когда реальные ядра закончились и начали использоваться виртуальные, производительность начала «скакать». Вызвано это, скорее всего, и так хорошей оптимизацией, в результате которой неиспользуемых вычислительных блоков для виртуальных ядер попросту не осталось. Не совсем понятные проблемы начались и у 48-ядерной системы от AMD, когда добавление ещё одного ядра резко замедляло вычисления. Логично предположить, что загвоздка как раз кроется в 48 ядрах — система сложная, и кто-то кому-то «не понравился».

<***SubTitle***> А победит дружба </***SubTitle***>

Подводя итог, стоит напомнить, что тесты проводились только на одной задаче. Если взять более требовательную к памяти проблему (как это делали в IBM), то видеокарты резко теряют позиции. Если использовать модификацию алгоритма Монте-Карло, то замедлятся уже центральные процессоры. Поэтому перспективной видится идея настоящих гетерогенных процессоров, предложенная в AMD под названием Fusion. В них на одном кристалле есть как ядра центрального процессора, так и графического ускорителя. И память у них общая. Поэтому если такие процессоры приживутся в суперкомпьютерах (а сейчас они только-только появились в мобильных устройствах), то проблемы выбора как таковой уже и не будет. «Тормозят» x86-совместимые ядра? Переключим вычисления на потоковые ядра GPU. Или наоборот. И делается это всё автоматически, возможно, даже без участия программиста.

</***Text***>

<***Таблица***>

Label: Тестовые системы

Тип системы	Количество ядер	Частота, *GHz*	Пиковая производительность, *GFlops*	Достигнутая производительность, *GFlops*	Эффективность
8 x AMD Opteron 8431	48	2,4	345,6	165,8	47,00%
4 x Intel Xeon X7450	24	2,4	230,4	130,5	56,00%
2 x Intel Xeon X5670	12 + 12	2,93	140,6	86,2	61,00%
2 x AMD Opteron 2427	12	2,2	79,2	39,6	50,00%
1 x NVidia Tesla C2050	448	1,1	1030,4	467,3	45,00%
1 x NVidia Tesla C1060	240	1,3	933	327	35,00%

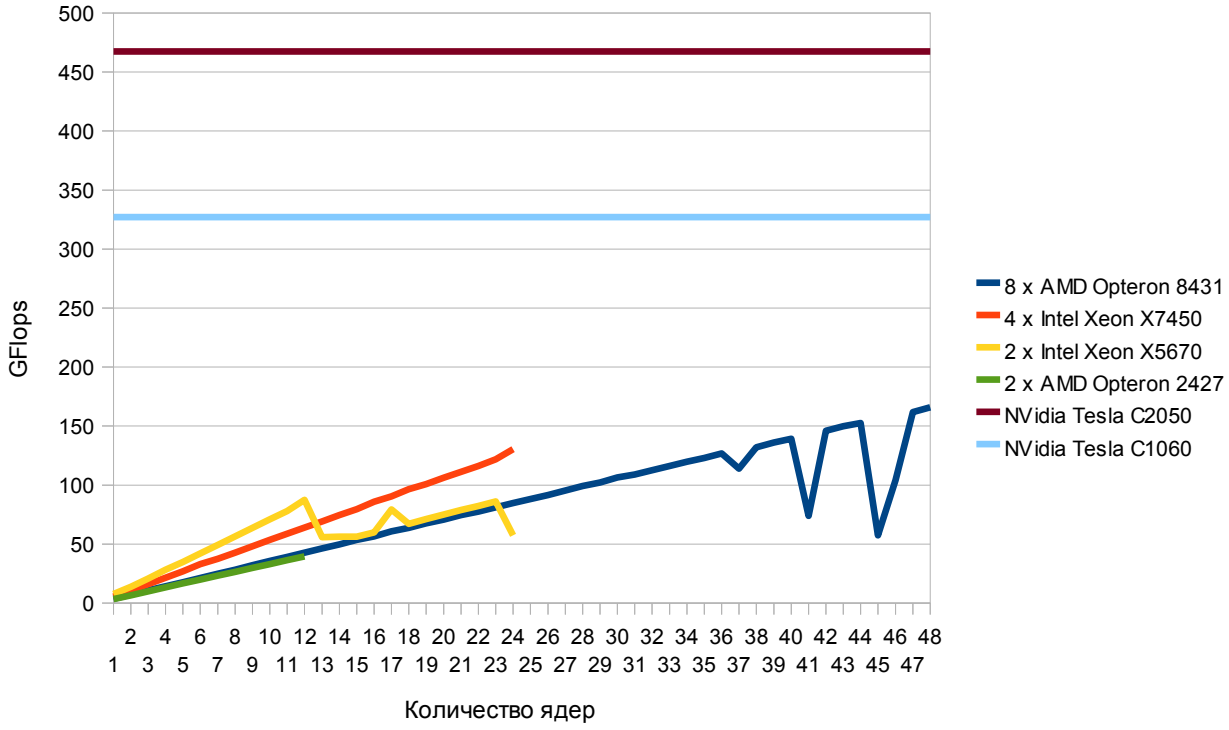
</***Таблица***>

<***Таблица***>
Label: Эффективность CPU-оптимизаций

Тип системы	Последовательный код	Intel C++	OpenMP	SSE	Всё вместе
8 x AMD Opteron 8431	1x	4x	35,9x	7,3x	267x
4 x Intel Xeon X7450	1x	5,2x	23,3x	8,9x	210,5x
2 x Intel Xeon X5670	1x	4,2x	11x	7x	82x
2 x AMD Opteron 2427	1x	4x	15x	7,3x	66x

</***Таблица***>

<***График***>



Label: И всё-таки CPU не удалось обойти GPU
</***График***>
<***Врезка***>
Бытует мнение, что Intel C++ является самым быстрым компилятором, но на процессорах Intel он ещё быстрее. Наши тесты это подтвердили.
</***Врезка***>