<title>

CUDA 4: покорение суперкомпьютеров

</title>

<hint>

В последний день зимы NVidia анонсировала новую версию CUDA, имеющую все возможности провести революцию в GPGPU. При этом в нововведениях заметен явный акцент на гетерогенные кластера. Станет ли CUDA SDK обязательным требованием для всех суперкомпьютеров? Удастся ли OpenCL продолжить сопротивление? Об этом и пойдёт речь.

</hint>

<text>

Сразу стоит оговориться, что под страшным словом «CUDA» обычно понимают две вещи: программную библиотеку и архитектуру видеокарты. Обе эти CUDA зависят друг от друга, но при этом имеют разную нумерацию. Так, последней версией «железа» является нашумевшая архитектура Fermi под номером 2.0, в то время как первой заточенной под неё библиотекой стал CUDA Toolkit 3.0.

Собственно, обновление, получившее известность как CUDA 4.0, коснулось как раз программной части. Подобные обновления выходят раз в 3-6 месяцев, радуя программистов новыми возможностями и несовместимостью с предыдущей версией (в результате чего приходится бросать все дела и срочно выяснять, почему же рабочая программа вдруг перестала компилироваться или запускаться). Глобальным событием это сложно было бы назвать, если не масштабность: NVidia улучшила функциональность существующих инструментов разработки, упростила модель программирования и внедрила GPGPU-аналог для библиотеки *STL*. Той самой, которая *Standard Template Library* для C++. Как следствие, получившийся «монстр» оказался самодостаточным — почти любую задачу можно решить без использования сторонних библиотек и инструментов.

Дружба с MPI

Не секрет, что одновременная работа с несколькими графическими ускорителями — занятие явно не для ленивых. Для каждого требуется создать свой поток, инициализировать CUDA-контекст, а потом ещё заниматься синхронизацией. В результате набрала популярность связка CUDA+MPI, ставшая фактически «народным решением». Она и проста в реализации, и масштабируется как на систему с несколькими ускорителями, так и на целый кластер. Есть только одна проблема — в ней сложно эффективно использовать центральный процессор (в котором, стоит заметить, найдётся пара сотен гигафлопс).

Собственно, эту проблему и решили в CUDA 4.0. Теперь с разными устройствами можно работать из одного CPU-потока, что идеально подходит для паттернов типа «мастер-рабочие». Но только этим дело не ограничилось - к одному GPU теперь можно обращаться из разных потоков. Если учесть, что ускорители с архитектурой Fermi поддерживают одновременное выполнение различных ядер, то открываются перспективы использования CUDA, например, совместно с директивами OpenMP. Будет ли от этого какая-либо практическая польза, сказать сложно, но жизнь программистов точно упроститься.

Также стоит упомянуть про заявленную официальную «интеграцию с MPI». Как следует из анонса, будет выпущена специальная модификация OpenMPI, адаптированная для работы с CUDA. В частности, с помощью операций Send/Receive можно будет копировать данные напрямую в видеопамять. Если вспомнить о революции в Top500, устроенной в 2010 году гетерогенными кластерами, то усилия NVidia становятся вполне понятными — чем проще использовать гетерогенный суперкомпьютер, тем быстрее CUDA станет «стандартнее» и так не очень популярного OpenCL.

Больше быстрой памяти

Самым заметным изменением в CUDA 4.0, с которым точно столкнётся большинство CUDA-разработчиков, стал переход к единому адресному пространству. Теперь память центрального процессора и всех графических ускорителей можно адресовать одним единственным указателем. Правда, чуда не произошло — память осталась физически разнородной и копирование «туда-сюда» никто не отменял. Зато теперь можно больше не задумывать о типе самого копирования (CPU→GPU или же GPU→CPU, а может GPU→GPU?) и смело забыть про страшные флаги вроде *cudaMemcpyHostToDevice*.

Другим нововведением стала технология NVidia GPU Direct 2.0, которая «обеспечивает равноправную связь между GPU в рамках одного сервера». Идея достаточно простая — при копирования из GPU-памяти всегда используется центральный процессор, что существенно тормозит весь процесс. Если узким местом оказались как раз подобные пересылки, то CUDA-программисту предлагается бесплатный комплект «бубнов» из различных классов памяти, специализированных аллокаторов и асинхронной пересылки данных. Если программист достаточно опытен, то он сможет ускорить копирование раза в 2-3. Если нет — то ему на помощь как раз и приходит технология GPU

Direct. Она позволяет копировать данные на GPU практически с любого устройства — будь то жёсткий диск, интерфейс InfiniBand или другой графический ускоритель. Заявляется, что CPU при этом не используется, что «повышает производительность на 30%». Правда, остаётся вопрос, чем версия GPU Direct 2.0 отличается от просто GPU Direct, вышедшего в начале 2010 года, но об этом NVidia пока предпочитаем умалчивать. Будем надеяться, что простым ребрендингом дело не ограничилось.

CUDA без CUDA

Другое направление, которое активно развивается благодаря усилиям NVidia — это библиотеки-надстройки над CUDA. Сначала появилась CUDA-версия BLAS, получившая название cuBLAS, потом была сделана качественная реализация Быстрого Преобразования Фурье (cuFFT), потом пришла очередь библиотеки для работы с разреженными матрицами (cuSPARSE) и генератора случайных чисел (cuRAND). Независимо разрабатывалась реализация LAPACK на CUDA (не угадали, не cuLAPACK, а просто CuLa) и библиотека компонент-кирпичиков, содержащая базовые алгоритмы обработки изображений (здесь обошлись даже без префикса «cu» - NPP).

С выходом новой версии CUDA, к существующим официально поддерживаемым библиотекам добавится OpenSource-проект Thrust. Его цель достаточно проста — создать аналог STL для CUDA. Если требуется отсортировать массив, то теперь не обязательно даже знать технологию CUDA, достаточно вызвать *thrust::sort*. Всё это вместе позволяет любому программисту создавать эффективные GPU-приложения без использования низкоуровневых вещей типа копирования памяти, запуска ядер, синхронизации. Другими словами, 28 февраля 2011 года каждый программист сам того не зная превратился в CUDA-программиста. Но скоро он об этом узнает, так как «маркетинговая машина NVidia» уже заработала.

Стоит заметить, что теперь практически для каждого программного инструмента Intel у NVidia есть свой аналог. Intel MKL покрывается связкой cuBLAS + Cula, Intel TBB заменяется новым Thrust, а для перехода от IPP к NPP достаточно заменить первую букву. Аналогичная ситуация и в средствах разработки — платная Intel Parallel Studio может быть заменена уже бесплатным NVidia NSight, а для утилит типа Intel VTune есть ответ в виде NVidia Visual Profiler.

Есть ли альтернатива?

Достаточно много надежд было возложено на GPGPU-стандарт OpenCL, призванный унифицировать гетерогенное программирование. Если же проследить за развитием событий, то ситуация оказывается достаточно грустной — идеи исследовательских проектов по расширению OpenCL в большинстве случаев уже реализованы в CUDA как коммерческие решения, а популярность этих двух технологий (если верить статистике запросов в Google.com), соотносится как 1:10. Более того, гиганты типа Adobe и MathWorks открыто перешли на сторону NVidia, начав использовать в своих продуктах CUDA.

С 2010 года NVidia начала активную экспансию на суперкомпьютерную индустрию, одним из проявлений которой и стал выход CUDA 4.0. Если ей удастся сохранить темп, то сейчас самое время начать изучать будущий стандарт программирования для гетерогенных систем. Даже если Intel и AMD не начнут поддерживать CUDA, то появятся сторонние решения, позволяющие запускать CUDA-программы на произвольных устройствах. К примеру, уже существует компилятор Ocelot, “на лету” транслирующий бинарную CUDA-программу в аналог для OpenCL или многоядерного процессора. Если этот проект постигнет такая же судьба, как и Thrust, то где-нибудь через год CUDA Toolkit 5.0 будут использовать даже владельцы не-NVidia ускорителей.

</text>