О МОДЕЛЯХ И ТЕХНОЛОГИЯХ ПРОГРАММИРОВАНИЯ СУПЕРКОМПЬЮТЕРОВ С НЕТРАДИЦИОННОЙ АРХИТЕКТУРОЙ.

С.С. Андреев, А.А. Давыдов, С.А. Дбар, А.О. Лацис, Е.А. Плоткина

Способ построения высокопроизводительных вычислителей путем объединения большого числа универсальных процессоров сравнительно слабыми каналами связи был основным на протяжении более 20 лет. Сегодня рост числа транзисторов на процессорном кристалле привел к тому, что способ этот в качестве основного изжил себя. Построенные таким путем вычислители обладают большим количеством «узких мест» системного характера, их к.п.д. на реальных задачах уже составляет первые проценты и продолжает быстро падать по мере совершенствования элементной базы. Катастрофическое падение к.п.д. наблюдается уже на уровне отдельного процессора [1]. Попытка решения проблемы чисто силовым путем, за счет линейного масштабирования многопроцессорной системы, приводит не только к еще большему падению к.п.д. и деградации параллельной эффективности, но и к недопустимому росту энергопотребления, трудно разрешимым проблемам с отводом тепла, не говоря уже о естественных проблемах с надежностью сверхбольших систем.

Все это вынуждает разработчиков высокопроизводительных вычислительных систем искать принципиально новые способы организации вычислителя, то есть разрабатывать новые архитектуры суперкомпьютеров. К сожалению, программировать для таких машин также приходится по-новому, причем в гораздо большей степени по-новому, чем 20 лет назад при переходе от последовательных машин к параллельным.

Переход к непривычным моделям и технологиям программирования крайне мучителен для прикладных программистов, но в среднесрочной перспективе ему, скорее всего, нет альтернативы. Задача разработчиков суперкомпьютеров в этих условиях видится в том, чтобы максимально структурировать работу по адаптации приложений, дать возможность прикладным программистам «преодолеть пропасть в два (хотя бы) прыжка».

Один из способов такого структурирования – построение машин новой архитектуры как гибридных [2]. Организующий каркас гибридной машины – многопроцессорный кластер на базе традиционных процессоров. Собственно нетрадиционная вычислительная мощность представлена в виде небольших сопроцессоров – ускорителей, включаемых в состав каждого вычислительного узла.

Гибридное построение системы позволяет сделать «первый из серии прыжков через пропасть» гораздо менее болезненным, чем в случае большого монолитного вычислителя новой архитектуры. Ключ к успеху в разработке гибридного приложения – в локализации на ускорителях тех фрагментов вычислительной процедуры, в которых интенсивность вычислений сочетается с максимально возможной локальностью обращения к памяти (быстрая, способная обеспечить работой сотни арифметических устройств одновременно, память в составе ускорителя – всегда ограниченный и дефицитный ресурс). Именно гибридное построение системы позволяет выполнить глубокую структурную перестройку программного кода, выделение пригодных к переносу на ускоритель фрагментов (вычислительных ядер) в терминах традиционного программирования, не затрагивая специфических понятий новых архитектур. Переработка программы таким способом не только не зависит от конкретной новой архитектуры, которую предполагается использовать впоследствии, но и значительно повышает быстродействие при работе на традиционном кластере из многоядерных процессоров. Работа эта может быть выполнена должным образом только прикладным программистом, хорошо знающим свой алгоритм. Она не имеет ничего общего с гипотетическим «превращением математиков в системщиков и электронщиков», которого обычно так боятся прикладные программисты, когда речь заходит о новых архитектурах. Кроме того, гибридное построение системы имеет вполне очевидные преимущества с точки зрения удешевления и упрощения создания новых суперкомпьютеров.

Есть у гибридного подхода и недостатки. Важнейший из них – значительное увеличение нагрузки на систему коммуникаций. К обычным для многопроцессорных машин коммуникациям между вычислительными узлами добавляются коммуникации внутри узла, между процессором и сопроцессором. Коммуникации – важнейший источник накладных расходов. Если их стало больше, а логика их организации усложнилась, то цену отдельного акта коммуникации надо снижать, а само программирование коммуникаций – упрощать. Возникает необходимость в разработке сетей межузловых коммуникаций, оптимизированных с точки зрения задержек и латентности. Такие сети полезны и сами по себе, безотносительно к гибридным машинам, но необходимость строить гибридные машины делает их разработку и использование еще более актуальными. К счастью, некоторый опыт разработок такого рода (и их поддержки на уровне системы параллельного программирования) в мировом сообществе разработчиков суперкомпьютеров имеется. Разработка специальных низколатентных сетей с возможностью поддержки параллельного программирования в стиле PGAS издавна практиковалась в фирмах Cray и SGI. Признавая очевидные преимущества таких сетей как с точки зрения увеличения эффективности распараллеливания «плохих» алгоритмов, так и с точки зрения упрощения такого распараллеливания, в большинстве своем пользователи суперкомпьютеров старались все же обходиться без них, поскольку стоили эти сети очень и очень дорого. Сейчас ситуация изменилась – построить «свой Cray» может

каждый, и будет такая машина не намного дороже кластера. Подробное обоснование этого тезиса, как и исследование вопроса о наиболее подходящих для таких сетей технологий параллельного программирования, выходит за рамки настоящего доклада. Отметим лишь, что низколатентная сеть с аппаратной поддержкой PGAS была разработана ИПМ им. М. В. Келдыша совместно с ФГУП «НИИ «Квант», и успешно используется в ИПМ в составе гибридного суперкомпьютера «МВС – Экспресс» [2].

Все рассказанное выше свидетельствует о том, что проблему собственно программирования вычислителей с нетрадиционной архитектурой при грамотной организации машины можно изолировать и минимизировать, освободить от сопутствующих дополнительных трудностей. Но, даже в локализованном и минимизированном виде, проблема программирования отдельных сопроцессоров – ускорителей остается чрезвычайно сложной. Разработка компиляторов традиционных языков программирования, способных по последовательной программе породить эффективный код для ускорителя – задача, если и разрешимая в принципе, то долгосрочная. В среднесрочной перспективе необходимы новые модели программирования, соответствующие языки и технологии.

Ситуация дополнительно осложняется тем, что сопроцессоры – ускорители бывают не только разных производителей и моделей, но и принципиально разной архитектуры. Выше мы отмечали, что предварительная работа по подготовке приложения к переносу на гибридную машину инвариантна относительно типа ускорителя. Можно ли сказать то же самое о новых моделях программирования самих сопроцессоров?

Наш, пока еще скромный, опыт использования разнотипных ускорителей для эффективной реализации вычислительных ядер [3] позволяет предварительно ответить на этот вопрос отрицательно. Ускорители разных архитектур отличаются друг от друга никак не слабее, чем каждый из них – от традиционного процессора. Отличия имеют качественный характер, для их учета модели программирования также должны отличаться качественно.

В последние 2 – 3 года большую популярность приобрели ускорители на базе видеокарт – GPGPU [4]. Различные GPGPU сильно отличаются по архитектуре, но все они построены по принципу программного управления ограниченным набором универсальных арифметических устройств. Уже такая степень общности позволила разработать для практически всех выпускаемых сегодня и перспективных GPGPU единую модель программирования, представленную платформенно-независимым языком OpenCL [4]. Казалось бы, усилия разработчиков программного обеспечения для других типов ускорителей должны были быть направлены на реализацию OpenCL для соответствующего оборудования. К сожалению, наш опыт разработки средств прикладного программирования для процессоров одной задачи в FPGA на сегодня показывает, что это не целесообразно. Процессоры одной задачи на базе FPGA вполне конкурентоспособны в сравнении с GPGPU, но только при использовании совершенно других типов параллелизма. В случае GPGPU параллелизм преимущественно векторно-мультитредовый, в случае FPGA – конвейерный, с очень большой глубиной конвейера, иногда – векторно-конвейерный. Для использования этих типов параллелизма реализуемая вычислительная процедура должна быть записана принципиально разными способами.

Отсутствие «общего знаменателя» в разработке систем программирования для GPGPU и FPGA – крайне плохая новость для прикладных программистов. Очевидно, что поиск этого «общего знаменателя» - открытая, притом крайне актуальная научно-техническая проблема. Искать ее решение можно с двух сторон. Можно пытаться найти модель, равно пригодную для эффективной реализации в обоих случаях. Можно пытаться найти промежуточный уровень организации схемы в FPGA, который позволил бы сблизить парадигмы представления вычислительной процедуры. До сих пор все поиски такого рода заканчивались потерями производительности примерно на порядок [3], по сравнению с реализацией в FPGA, выполненной подходящими инструментами. В докладе подробнее рассказывается о попытках обобщения идеи VLIW-процессора, который мог бы претендовать на роль эффективного промежуточного уровня организации FPGA. Кроме того, кратко анализируется проблема унификации подходов к учету иерархии типов внутренней памяти для разнотипных ускорителей.

ЛИТЕРАТУРА:

1. В. В. Воеводин. Суперкомпьютеры и парадоксы неэффективности. «Открытые системы», №10, 2009г.
2. С. С. Андреев, А. А. Давыдов, С. А. Дбар, А. Б. Карагичев, А. О. Лацис, Е. А. Плоткина. Макет гибридного суперкомпьютера МВС-экспресс. Тезисы докладов 17-й Всероссийской конференции «Теоретические основы и конструирование численных алгоритмов и решение задач математической физики с приложением к многопроцессорным системам», Дюрсо, 2008г.
3. Андреев С. С., Дбар С. А., Лацис А. О., Плоткина Е. А. Система программирования Автокод HDL и опыт ее применения для схемной реализации численных методов в FPGA. Тезисы докладов Всероссийской конференции «Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность», Дюрсо, 2009г.
4. www.nvidia.com. Сетевой ресурс.