РАЗВИТИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КОММУНИКАЦИОННОЙ СИСТЕМЫ МВС-ЭКСПРЕСС И НЕКОТОРЫЕ ПРОБЛЕМЫ РАЗРАБОТКИ ГИБРИДНЫХ ПРИЛОЖЕНИЙ ДЛЯ СУПЕРКОМПЬЮТЕРА К-100

С.А. Дбар, Ю.А. Климов, М.М. Краснов, А.О. Лацис, В.Л. Орлов, Г.П. Савельев, А.А. Соколов, М.Ю. Храмцов, Вик.С. Штаркман

Суперкомпьютер К-100 [1], работающий в ИПМ РАН уже почти 2 года, отличается от традиционного вычислительного кластера в двух отношениях. Во-первых, это гибридная вычислительная система: в состав вычислительного узла входят 2 универсальных процессора и 3 ускорителя вычислений GPGPU Nvidia Fermi C2050. Во-вторых, К-100 оснащен сетью «МВС-Экспресс» (совместная разработка ИПМ им. М. В. Келдыша, ФГУП «НИИ «Квант» и ООО «НПО «Роста») [2]. Проблемы сети межузловых коммуникаций в многопроцессорных системах и проблемы программирования гибридных вычислительных узлов часто рассматриваются по отдельности. Опыт эксплуатации К-100 позволяет нам поставить эти две группы проблем в общий контекст проблематики разработки гибридных суперкомпьютерных приложений. Чтобы понять единство и глубинное родство двух упомянутых групп проблем, рассмотрим их сначала традиционным образом – по отдельности.

Сеть «МВС-Экспресс» задумывалась несколько лет назад как попытка обеспечить узлы вычислительного кластера системой общей памяти на аппаратном уровне. Предполагалось, что решение этой задачи само по себе обеспечит прорыв в упрощении параллельного программирования, в повышении его продуктивности. С одной стороны, казалось, что самое трудное – это обеспечить саму возможность доступа одного процессора к памяти другого обычными командами обращения к памяти, а все остальные «несущественные технические вопросы» решатся почти сами собой. С другой стороны, упомянутая возможность с появлением магистрали PCI Express оказалась буквально лежащей на поверхности: для прямого соединения PCI Express двух компьютеров между собой требовалось совсем немного очень простого оборудования. Быстро выяснилось, что соответствующие микросхемы выпускаются серийно. Почти сразу было выпущено несколько пробных комплектов сетевого оборудования. Оставалось разработать программное обеспечение, более привлекательное для прикладного программиста, чем MPI, и предложить его реальным пользователям.

Не так быстро, как хотелось бы, но все основные технические проблемы низкоуровневой настройки единой для всего кластера, комбинированной магистрали PCI Express были решены. В физическом адресном пространстве каждого вычислительного узла появились окна доступа к выделенным областям физической памяти всех прочих узлов, причем без каких-либо искусственных ограничений на размер и количество таких окон. Пришло время разработки конкретного API параллельного программирования.

Наиболее привлекательным для прикладного программиста представлялся «наивный» стиль использования получившейся асимметричной (NUMA) общей памяти, примерно такой: «Каждый процесс обращается к общей памяти, но следит за тем, чтобы обращений к близким ему областям такой памяти было в среднем гораздо больше, чем к далеким». К сожалению, такой режим в рамках МВС-Экспресс оказался не реализуемым. Дело в том, что он подразумевает довольно интенсивное чтение «далеких» областей общей памяти, что, в свою очередь, не может быть эффективно реализовано без ее кэширования. Кэширование же окон доступа в «чужую» память в рамках архитектуры МВС-Экспресс оказалось невозможным на уровне чипсетов современных процессоров. Доступ же в «чужую» память без кэширования запредельно неэффективен даже тогда, когда происходит сравнительно редко.

С другой стороны, очень эффективным получился режим произвольного, мелкозернистого доступа к «чужим» областям общей памяти на запись, но не на чтение. Как хорошо известно, в том числе, из дискуссий на прошлых конференциях, такому соотношению эффективностей чтения и записи соответствует не упомянутый выше «наивный» стиль, а, скорее, модель односторонних коммуникаций типа shmem. О преимуществах этой модели перед MPI, о ее способности в большей степени раскрыть перед пользователем преимущества современных сетей, сказано и написано немало [3]. Некоторый диалект shmem был реализован для сети МВС-Экспресс, объявлен базовым API и в таком качестве предложен пользователям.

Довольно быстро выяснились два малоутешительных экспериментальных факта.

Во-первых, разработчики современных версий InfiniBand также реализовали shmem на своем оборудовании [4].

И, во-вторых, имея в своем распоряжении на К-100 две качественных реализации shmem, прикладные программисты не используют ни одной, предпочитая MPI. Преимущества новой модели программирования оказываются недостаточными, не дают нового качества, не «перевешивают» преимуществ стандартности традиционной технологии. Возникает резонный вопрос:

А зачем, в таком случае, вообще нужна сеть МВС-Экспресс?

Разработчики собственных коммуникационных систем часто обосновывают необходимость своих разработок их хорошей масштабируемостью до рекордных размеров. МВС-Экспресс этим похвастаться не может. Правда, МВС-Экспресс заметно превосходит InfiniBand по таким показателям, как латентность и, в особенности, темп выдачи сообщений. Это легко заметить по тестам, а также по скорости выполнения специально подобранных модельных задач. Но пользователи крайне редко решают подобные задачи на практике, а если и решают – то не используют shmem. Как же конвертировать видимые только на тестах, и понятные только разработчикам, преимущества сети в нечто осязаемое, не просто понятное пользователю, но и привлекательное для него?

Сегодня единственным API традиционного параллельного программирования, на практике доказавшим свою значительно большую привлекательность, чем MPI, является OpenMP. Если бы вновь создаваемое сетевое оборудование оказалось значительно более пригодным для реализации OpenMP, чем InfiniBand, то уже одно это стало бы более чем достаточным основанием для разработки такого оборудования. Вопросы же поддержки общепринятых сетевых протоколов, как и вопросы масштабируемости сети, в этом случае, безусловно, отошли бы на второй план.

Попытки построить реализации OpenMP на базе коммуникационной системы в рамках кластера предпринимались неоднократно[5], и приводили обычно к формально работоспособным, но крайне неэффективным системам [6]. Было решено исследовать сеть МВС-Экспресс на предмет выяснения, не могут ли ее преимущества в латентности и темпе выдачи сообщений послужить предпосылкой для эффективной реализации виртуальной общей памяти в масштабах кластера. Исследование предполагало реализацию не OpenMP целиком, а только необходимой для такой реализации «главной детали»: виртуальной общей памяти, доступной по машинному указателю, и когерентной с точностью до явной барьерной синхронизации.

Проектирование системы виртуальной общей памяти в виде библиотеки функций и набора макросов происходило в два этапа. На первом этапе было решено несколько сузить постановку задачи, то есть реализовать не «виртуальную SMP», а «виртуальную NUMA». Сделано это было для того, чтобы для некоторого, достаточно широкого, класса задач получить заведомый выигрыш в эффективности реализации уже на архитектурном уровне. Библиотека получила название RefNUMA (Reflective NUMA) [7]. На втором этапе библиотека была реализована на shmem. Исследование быстродействия RefNUMA на простых модельных задачах показало, что реализация с использованием shmem-Экспресс во многих случаях работает заметно быстрее, чем реализация с использованием InfiniBand shmem. При этом упомянутый выше стиль «наивного» использования общей памяти, с доступом к «чужим» областям преимущественно на чтение, а не на запись, поддержан в полной мере. Эффективность в целом сопоставима с эффективностью реализации тех же приложений при помощи MPI. Полученные результаты позволяют нам планировать полноценную реализацию OpenMP на базе RefNUMA в обозримой перспективе.

Теперь вернемся к самому началу наших рассуждений, и попробуем понять, как приведенные выше соображения на тему коммуникационной системы соотносятся с проблематикой разработки именно гибридных приложений.

Накопленный за время эксплуатации сначала – макетов и опытных образцов гибридных машин, а затем – собственно суперкомпьютера К-100, опыт позволяет нам сделать следующее, на первый взгляд – немного странное, утверждение. Сложность разработки гибридных приложений заключается, в первую очередь, вовсе не в необходимости использовать специальные средства программирования ускорителей, такие, как CUDA или Open CL [8]. Не меньше, если не больше, проблем вызывает необходимость явно выписывать в тексте своей программы сложную, вычурную механику согласованного перемещения данных между узлами кластера, с одной стороны, и между процессорами и ускорителями внутри узла, с другой. Так, в простейшем модельном приложении – решении задачи для двумерного сеточного аналога уравнения теплопроводности методом Якоби – типовой прием разбиения расчетной области на блоки с теневыми гранями приходится применять трижды! Сам по себе этот прием весьма прост, но необходимость выписывать его многократно разными способами утомляет программиста, повышает вероятность ошибок и затрудняет отладку. В этих условиях всякое заметное упрощение записи коммуникаций вносит весьма ощутимый вклад в упрощение разработки гибридного приложения в целом. Легко показать, что технология RefNUMA, при ее дальнейшем совершенствовании, позволяет автоматизировать, скрыть от прикладного программиста не только коммуникации между вычислительными узлами как таковые, но и заметную часть тех коммуникаций между процессором и ускорителем, которые возникают в связи с необходимостью межузловых коммуникаций в гибридной многопроцессорной вычислительной системе. Запись гибридной параллельной программы при этом становится похожей на запись гибридной программы для одного процессора, оснащенного одним ускорителем. Это позволяет нам рассматривать технологию RefNUMA в среднесрочной перспективе как еще один подход к разработке и реализации языков гибридного параллельного программирования.

ЛИТЕРАТУРА

1. К-100 // www.kiam.ru Сетевой ресурс.
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. А. А. Корж. Graph500: обгоняя на повороте // Журнал «Суперкомпьютеры», №3 (7), осень 2011г., с. 44-46.
4. http://filedownloads.qlogic.com/files/driver/73591/QLogic_SHMEM_UG_[B].pdf Сетевой ресурс.
5. Intel Cluster OpenMP User’s Guide // http://software.intel.com/file/6330 Сетевой ресурс.
6. А. О. Лацис. Параллельная обработка данных // Издательский центр «Академия» Москва, 2010г. 336с. ISBN 978-5-7695-5951-8.
7. RefNUMA // http://www.kiam.ru/MVS/documents/k100/refnumaprogman.html Сетевой ресурс.
8. Nvidia // www.nvidia.com Сетевой ресурс.