Андреев С. С., Дбар С. А., Давыдов А. А., Лацис А. О., Савельев Г. П., Орлов В. Л., Плоткина Е. А., Простов И. В.

Гибридный суперкомпьютер К-100: что дальше?

Не прошло и трех лет с тех пор, как гибридные системы на базе GPGPU прочно заняли первые позиции в Top-500 [1]. Суперкомпьютерный мир в очередной раз очень быстро и внезапно стал другим. Какие выводы можно уже сейчас сделать из этого технологического переворота? Каков прогноз на ближайшую перспективу? Над чем работать разработчикам, к чему готовиться пользователям?

1. Что в действительности произошло.

На первый взгляд, в силу стечения ряда обстоятельств на рынке просто появилась новая технология (GPGPU), которую стало модно использовать в высокопроизводительных вычислениях. В суперкомпьютерном сообществе сегодня преобладают два, казалось бы, взаимно противоположных взгляда на эту технологию. Согласно первому, будущее, несомненно, за «суперкомпьютерами на видеокарточках», согласно второму – все скоро успокоится, «мода» пройдет, суперкомпьютеры снова будут строиться на традиционных многоядерных процессорах. Легко заметить, что в этих двух позициях, в действительности, гораздо больше сходного, чем различного. Сторонники обеих молчаливо исходят из того, что суперкомпьютеры и дальше будут строиться некоторым единым способом, который установится на длительное время. Спор идет лишь о том, что именно за способ это будет.

Очень похожие ситуации, сопровождающиеся очень похожими спорами, в истории техники случались уже не раз. Например, в первой половине 19 века до хрипоты спорили о том, какое средство транспорта для путешествий на большие расстояния будет отныне преобладать – парусник или все же колесный пароход на дровяной тяге? Сегодня, почти 200 лет спустя, мы прекрасно понимаем, что дилемма эта была ложной. Колесный пароход ни в коей мере не мог служить заменой паруснику, который до этого был, практически, единственным средством дальних перемещений на протяжении тысячелетий. Чем он был в действительности – так это первой более или менее удачной попыткой найти паруснику альтернативу. Сам факт этой первой попытки совершенно точно свидетельствовал о том, что за ней неизбежно и скоро последует вторая, третья и так далее. В результате этой серии попыток, как мы знаем сегодня, не просто произошла замена одного типа судна (парусник) каким-то другим. Вместе с парусным флотом ушло в прошлое само единообразие технических решений – современные морские и воздушные суда невообразимо разнообразны. Лишь некоторые типы современных кораблей отдаленно напоминают колесный пароход. Парусниками же пользуются только спортсмены и отдыхающие миллионеры.

Современная ситуация в суперкомпьютерной отрасли представляется во многом аналогичной. «Мода на видеокарточки», конечно, скоро пройдет, порожденные этой «модой» технологии займут свою нишу и перестанут рассматриваться как панацея. При этом довольно наивно было бы полагать, что все вернется к «старому доброму универсальному процессору». Причины,

заставившие разработчиков искать принципиально новые архитектуры суперкомпьютеров, хорошо известны и неоднократно обсуждались [2-5]. Они имеют системный характер, их корни неизмеримо глубже, чем любая отдельная технология или, тем более, чем любая «мода». Возврата назад не будет – он попросту невозможен, хотя бы по причинам хорошо известных энергетических ограничений. Да, первый номер в Green-500 сегодня занимает суперкомпьютер на универсальных процессорах [6]. Но ведь и первые пароходы плавали медленнее чайных клиперов…

Основные выводы из сказанного в этом разделе могли бы звучать так:

1). Технология GPGPU, безусловно, не является технологическим курьезом, ее появление глубоко закономерно. Это так же верно, как и то, что она не может претендовать на роль универсальной замены традиционного процессора. Технологию эту следует рассматривать как «конструктивное доказательство теоремы существования», как первую и, несомненно, удачную в целом попытку создать альтернативную архитектуру суперкомпьютера. Как бы ни сложилась дальнейшая судьба этой первой попытки, за ней последуют другие, и их будет много, поскольку возврат назад представить себе нелегко. Предсказать конкретный облик новых архитектур сегодня крайне трудно, но уже совершенно ясно, что они будут, причем будут разными, и очень скоро.

2). Реальной, а не вымышленной, технологической дилеммой является не выбор между универсальным процессором и GPGPU, а выбор между системами массового параллелизма на базе «легких» или же «тяжелых» вычислительных узлов.

Подход на базе «легких» вычислительных узлов состоит в экстраполяции традиционной кластерной технологии на современной технологической базе, и предполагает, с учетом необходимых уже завтра уровней суммарной производительности, построение однородных вычислительных структур из миллионов процессорных ядер. Такие системы, при всем их формальном сходстве с традиционными вычислительными кластерами, потребуют совершенно новых подходов к программированию приложений, которые еще только предстоит разработать.

Альтернативный подход на базе «тяжелых» вычислительных узлов предполагает построение систем из всего лишь тысяч гибридных узлов, в каждом из которых под управлением небольшого числа процессорных ядер сосредоточена сильно связанная вычислительная мощность нетрадиционной архитектуры. Технология GPGPU-кластеров – ставший уже хрестоматийным пример подхода на базе «тяжелых» узлов, и пример, безусловно, работающий. Аналогичного по степени освоенности в реальной производственной практике подхода на базе «легких» узлов пока не просматривается. Мы полагаем, что среднесрочное, по крайней мере, будущее – за системами из «тяжелых» вычислительных узлов. Конкретная «нагрузка» этих узлов, конечно же, будет меняться.

3). Необходимость программировать гибридные вычислительные узлы с сопроцессорами нетрадиционной архитектуры ставит как перед разработчиками, так и перед пользователями суперкомпьютеров целый ряд невообразимо сложных задач. Об этом – ниже.

2. Несколько слов о суперкомпьютере К-100.

В данном разделе мы не стремимся представить краткий обзор К-100. Всю необходимую информацию о К-100 можно получить, например, на нашем сайте [7]. Здесь же хотелось бы поговорить о том, какие подходы применялись при строительстве этой машины и почему.

Появлению К-100 предшествовал примерно трехлетний этап строительства макетов и опытных образцов. Начиная эту работу, мы в общих чертах понимали практически все из сказанного выше, и относились к нашим макетам именно как к «конструктивному доказательству теоремы существования». Понимая исключительную важность и актуальность поиска новых суперкомпьютерных архитектур, мы решили начать с той из них, которая обещала наиболее быструю практическую отдачу, не забывая при этом ни на минуту, что это – только частный случай, только первая попытка, что поиск «правильных» архитектур этой попыткой не заканчивается, а только начинается.

С самого начала работы мы хорошо понимали, что ориентация на «тяжелые» вычислительные узлы гибридной архитектуры выдвигает совершенно особые требования к системе межузловых коммуникаций. В гибридной вычислительной системе возрастают, по сравнению с традиционным вычислительным кластером, как нагрузка на систему коммуникаций, особенно в части задержек, так и сложность программирования этих коммуникаций в прикладных программах. Мы не только разработали свою коммуникационную сеть, но и постарались найти для нее подходящую технологию программирования, альтернативную MPI [8]. Сегодня отрадно сознавать, что наши исследования в этой области вполне вписались в мировой контекст. Когда К-100 уже строился, мы узнали о том, что фирма QLogic [9] – признанный лидер в мировом сообществе разработчиков сетей для суперкомпьютеров – вышла на рынок с предложением очень похожей сети, оснащенной в точности той же альтернативной технологией параллельного программирования, которую мы выбрали для нашей сети МВС-Экспресс. В результате, К-100 оснащен обеими упомянутыми сетями. Сейчас мы работаем над тем, чтобы сделать нашу сеть более масштабируемой.

Все макеты и опытные образцы GPGPU-кластеров, создаваемые в ИПМ им. М. В. Келдыша, оснащались той же системой управления ресурсами [10], что и кластеры, находящиеся в промышленной эксплуатации, и, благодаря этому, были доступны для первых пользователей в режиме 24x7. Это позволило нам не только провести испытания аппаратных и программных решений и, в конечном итоге, принять решение о строительстве К-100, но и подготовить, к моменту запуска К-100, группу пользователей, уже имеющих опыт практического применения гибридных машин, и умеющих отвечать на вопросы своих менее опытных коллег. Полагаем, что именно этим, не в последнюю очередь, объясняется полное отсутствие «проблем с поиском приложений» и, тем более, «проблем с загрузкой машины». Сегодня, через 10 месяцев после запуска в опытную эксплуатацию, К-100 работает в штатном режиме, и загружен в среднем более чем наполовину.

3. Проблемы разработки гибридных приложений.

Примерно трехлетний опыт поддержки пользователей GPGPU-кластеров позволяет нам сделать первые выводы об основных проблемах программирования таких машин.

Хорошо известно, что традиционные языки программирования для вычислителей нетрадиционной архитектуры не годятся [2,5]. Проблема не в лексике и синтаксисе языка – их как раз можно сделать в новом языке как угодно традиционными – а в самом наборе понятий, на которых строится язык, то есть в модели программирования. Этот бесспорный на сегодняшний день факт породил кажущееся очевидным, но, тем не менее, в корне неверное представление о том, что главная проблема применения новых машин состоит в необходимости изучения новых, непривычных языков.

Проблема освоения языков программирования нетрадиционных вычислителей, конечно, существует. Более того, именно сравнительно «слабая нетрадиционность» языков программирования GPGPU, их не очень глубокое отличие от языков традиционного типа, и обусловило во многом успех применения GPGPU-кластеров. И все же, как нам кажется, основная сложность разработки приложений для гибридных машин – не в необходимости программировать на непривычных языках. Основная сложность состоит в необходимости организовывать перемещение данных между памятью универсального процессора и различными видами памяти сопроцессора-ускорителя, согласуя эти перемещения с перемещением данных между вычислительными узлами.

Как неоднократно отмечалось еще при самых первых применениях суперкомпьютеров на базе «тяжелых» гибридных узлов [4], при переносе приложения на такую машину с традиционного вычислительного кластера оно должно быть подвергнуто глубокой структурной переработке, чтобы выделить явно те фрагменты кода и области памяти, которые должны обрабатываться на сопроцессоре-ускорителе. Если эта переработка первоначального варианта текста программы выполнена правильно, то часть программы, подлежащая переносу на ускоритель, оказывается простой и небольшой по объему, что делает ее разработку сравнительно несложной (300 строк простого кода можно написать с использованием любой модели программирования, или, в крайнем случае, попросить это сделать кого-то другого). Вся остальная часть программы – а это, скорее всего, более 90% ее исходного текста – остается программой для универсального процессора, и, казалось бы, ее разработка (доработка) не должна вызывать принципиальных проблем. В действительности же основные проблемы кроются именно в этой, остающейся на универсальном процессоре, части программы. Именно на нее ложится согласованное разделение работы на части между узлами кластера, с одной стороны, и на обрабатываемые ускорителем (ускорителями) порции, с другой. Именно на нее ложится необходимость согласовать со всем этим два вида коммуникаций и синхронизации – между вычислительными узлами, с одной стороны, и внутри вычислительного узла – с другой. Все это совершенно традиционная программистская работа, записываемая для традиционных процессоров на традиционных языках, но ее много, она нетривиальна и запутанна, в ней легко ошибиться, а найти ошибку – трудно. Ошибившись в этой «коммуникационно-организационной» части программы, передав в ускоритель немного не те данные, немного не в том количестве, и получив, естественно, немного не те результаты, программист склонен винить «особенности округления на GPGPU», «ошибки

транслятора» и прочие мифические сущности, редко встречающиеся в действительности. Так и не устранив проблему, он делает вывод о «сырости» технологии и «сложности программирования для этих хваленых видеокарточек», после чего может и перестать быть нашим пользователем.

Таким образом, проблема программирования для ускорителей нетрадиционной архитектуры как таковая существует, но первостепенной не является. На практике гораздо важнее проблема описания и отладки гетерогенных коммуникаций, порожденная гибридным характером машины.

Поскольку проблема эта инвариантна относительно конкретного вида сопроцессора-ускорителя, всякое продвижение в ее решении автоматически означает очень важный шаг к освоению любых гибридных машин, даже тех, которые еще не появились, а не только GPGPU-кластеров. Поскольку проблема эта – «чисто технического», а не «содержательного», плана, формулируется и должна решаться средствами традиционного параллельного программирования, у нас – разработчиков систем программирования для суперкомпьютеров – нет ни технических, ни моральных оправданий для уклонения от решения этой проблемы в самые короткие сроки.

Основное направление поисков решения – максимальное упрощение и автоматизация средств организации гетерогенных коммуникаций. Освобожденный от этого дополнительного и, по сути, совершенно не принципиального, но очень утомительного и неудобного груза, пользователь, скорее всего, уже сравнительно легко справится с основной сложностью содержательного характера – с программированием вычислителей нетрадиционной архитектуры как таковым.

Двигаясь в указанном основном направлении, нам – разработчикам систем программирования – практически ничего не придется изобретать. Все уже придумано довольно давно, и нуждается лишь в систематизации и адаптации к особенностям гибридной машины. За два десятилетия попыток как можно полнее автоматизировать традиционное параллельное программирование появилось много интересных, но мало используемых, частично забытых, технологий.

Еще на заре параллельного программирования появилась библиотека ScaLAPACK [11]. Основная идея, положенная в ее основу, состоит в том, что большинство параллельных программ используют лишь небольшое число типовых шаблонов обмена данными между вычислительными узлами, которые могут быть написаны раз и навсегда в виде коллективных операций. Пользуясь такими типовыми коллективными операциями, программист уже не рискует ошибиться на единицу в вычислении индекса в пересылаемом на другой вычислительный узел массиве – все вычисления индексов и подобная им техническая работа по организации обменов реализованы раз и навсегда и скрыты внутри библиотеки. Впоследствии это подход получил развитие в целом ряде похожих библиотек, наиболее известная из которых – PetsC [12].

В эпоху традиционных кластерных систем библиотеки проблемно-ориентированных операций коллективного обмена данными находились, в некоторой степени, на периферии работ по автоматизации параллельного программирования. В самом деле, они автоматизируют параллельную реализацию не произвольной, как угодно давно и плохо написанной, программы, а «всего лишь» процентов 60-70 приложений, которые можно отнести к типовым по способу организации обменов данными между узлами, да и то – не полностью автоматизируют, а только

помогают программисту быстро и без ошибок модифицировать программу. Для плохого программиста такая технология все еще слишком сложна, для грамотного и хорошо мотивированного – иногда проще все написать самому с нуля, чем изучать описание библиотеки. Когда машина становится гибридной, а коммуникации – гетерогенными, «написать с нуля» становится уже совсем не проще, и привлекательность такого инструмента для, условно говоря, хорошего программиста значительно возрастает. Она возрастет еще больше, если обобщить и развить подход применительно к гибридным коммуникациям, ввести и реализовать понятие типового гибридного коммуникационного шаблона.

Следует отметить, что сделанное выше замечание о нахождении такого рода технологий «на периферии работ по автоматизации параллельного программирования» в эпоху традиционных кластерных систем не вполне справедливо. С точки зрения тех, кто разрабатывал в то время средства автоматизации параллельного программирования, это замечание, пожалуй, верно. С точки же зрения пользователей, прикладных программистов, ситуация смотрится несколько иначе. По количеству практических применений среди высокоуровневых, отличных от ручного использования MPI, технологий, именно технология библиотек проблемно-ориентированных коллективных операций прочно занимает первое место.

Выше мы уже упоминали о том, что создание машин на «тяжелых» гибридных вычислительных узлах неразрывно связано с совершенствованием системы межузловых коммуникаций, с использованием очень хороших внутренних сетей кластера. Хорошая сеть позволяет также подвергнуть ревизии некоторые потенциально привлекательные коммуникационные технологии, изобретенные ранее, но не очень хорошо показавшие себя в прошлом, на использовавшемся тогда коммуникационном оборудовании. Так, на К-100 нам удалось выполнить макетную реализацию рефлективной общей памяти. Она пригодна не для всех классов параллельных программ, но гораздо более проста для программиста и эффективна в реализации, чем, например, такая известная реализация рефлективной общей памяти, как Intel Cluster Open MP [13].

4. А что же дальше?

Как мы уже отмечали выше, мировая суперкомпьютерная отрасль находится сегодня в интенсивном поиске новых вычислительных архитектур, среди которых GPGPU – лишь первая удачная попытка. С этой точки зрения, любой суперкомпьютер, который работает, и нашел своих пользователей, уже не интересен по определению. Возвращаясь к метафоре из первого раздела этой статьи, мы можем констатировать, что колесный пароход построен и приступил к регулярным рейсам. Кто, и как именно, теперь спроектирует и построит теплоход с гребным винтом, катер на подводных крыльях и самолет? Или, выражаясь более точно, каковы источники дальнейшего появления новых суперкомпьютерных архитектур? Можно ли, и как именно, участвовать в этом процессе?

Проще всего было бы сказать, что развитие массового, коммерческого сектора компьютерной промышленности, рано или поздно, приведет к возникновению некоторых новых «супер-GPGPU»,

в ожидании которых нам, разработчикам суперкомпьютеров, остается только шлифовать мелкие детали системного программного обеспечения. Нам представляется, что сегодня такая выжидательная тактика не оправдана по целому ряду причин.

Ситуация настолько напряженного, как сейчас, архитектурного поиска сложилась в мировой компьютерной индустрии впервые в ее истории. В результатах этого поиска заинтересована по-настоящему только отрасль высокопроизводительных вычислений – ничтожная, по коммерческим меркам, часть мировой компьютерной индустрии. Ни из чего не следует, что мировая компьютерная индустрия изделий массового выпуска способна выполнить за нас, разработчиков суперкомпьютеров, работу по созданию перспективных вычислительных архитектур. Один раз – в случае с GPGPU – нам просто повезло. Следует ли и дальше рассчитывать на везение, или поиск и разработку новых архитектур следует взять в свои руки?

Безусловно, их следует взять в свои руки, если только для этого есть техническая возможность. Такая возможность есть. Связана она с построением реконфигурируемых вычислительных систем на базе ПЛИС.

Идея ускорения вычислений за счет прямой схемной реализации вычислительных процедур в ПЛИС гораздо старше, чем технология GPGPU [14]. С появлением GPGPU – дешевого, выпускаемого серийно и сравнительно легко программируемого ускорителя вычислений – о реконфигурируемых сопроцессорах на базе ПЛИС говорят все реже. На первый взгляд, они потеряли свою актуальность. Но так обстоит дело только в том случае, если мы будем сравнивать, с учетом сегодняшней рыночной стоимости и сегодняшнего состояния технологий программирования, возможности создания готовых к немедленному промышленному применению вычислительных систем на GPGPU, с одной стороны, или же на ПЛИС – с другой. В этом случае, конечно, ПЛИС не в состоянии составить конкуренцию для GPGPU. Мы, однако, говорим не о немедленной постановке под рабочую нагрузку готовых систем, а об архитектурных исследованиях. Машины с «тяжелыми» гибридными узлами на базе ПЛИС, будучи оформлены как системы удаленного коллективного доступа, подобно традиционным кластерным системам, обладают уникальной возможностью, которой не может похвастаться ни один другой класс суперкомпьютеров. Они способны во много раз расширить круг создателей и испытателей новых вычислительных архитектур. Дать широкому кругу пользователей из суперкомпьютерного сообщества, причем вовсе не только тем, кто сегодня причисляет себя к разработчикам «железа», возможность создать и испытать собственноручно разработанное вычислительное оборудование, подобно тому, как пользователи традиционных кластерных систем разрабатывают и испытывают в удаленном режиме собственные программы. И, если даже всего лишь одна сотая таких самостоятельно разработанных схем окажется прообразом «новых супер-GPGPU», поможет промышленности лучше понять, чего именно хотят от нее пользователи суперкомпьютеров, работу по созданию и поддержке эксплуатации таких «архитектурных полигонов коллективного пользования» уже можно смело считать не пропавшей даром. Таким, и только таким, способом

суперкомпьютерное сообщество, включая не только разработчиков, но и пользователей, может реально взять задачу создания новых вычислительных архитектур в свои руки.

При этом, сделанное выше замечание о безнадежности конкуренции ПЛИС с GPGPU тоже не следует понимать слишком буквально. По нашим оценкам, современные ПЛИС примерно равноценны современным GPGPU по реально достижимой производительности, будучи значительно более «холодными». Так что реализация вычислений в ПЛИС уже сегодня вовсе не является заведомо проигрышной по эффективности, даже в сравнении с лучшими из GPGPU. Тем более, что рабочие частоты ПЛИС, в отличие от рабочих частот универсальных процессоров и GPGPU, пока еще довольно быстро растут.

ИПМ им. М. В. Келдыша недавно ввел в опытную эксплуатацию вычислительную систему РВС-КП, построенную НИИ «Квант» совместно с НПО «Роста». Это небольшой кластер с гибридными вычислительными узлами, в каждый из которых входит маломощный 2-ядерный процессор (Core 2 Duo) и 8 ПЛИС. Система оснащена штатной системой управления ресурсами [10], позволяющей использовать ее дистанционно, в режиме 24x7, и системой программирования ПЛИС собственной разработки [3], на базе схемотехнической САПР фирмы Xilinx [15]. Система разработки гибридных приложений полностью документирована [16], предоставляет пользователю возможность самостоятельно разрабатывать с нуля собственные аппаратно-программные вычислительные комплексы, причем без привлечения каких-либо дополнительных средств схемотехнического проектирования. Документация пользователя включает несколько примеров применения разной степени сложности. Для ее чтения и понимания не требуется не только предварительного знакомства с какими-либо схемотехническими САПР, но и вообще никакой специальной подготовки в области проектирования «железа». Достаточно базовых знаний и опыта в области программирования. В настоящее время мы заняты дальнейшим совершенствованием системы программирования, а также готовим ввод в строй еще одной похожей системы, также малоразмерной, но с более мощными процессорами и более современными ПЛИС.

Литература.

1. www.top500.org. Сетевой ресурс. Время доступа 11.00 4 октября 2011г.
2. Лацис А. О. Параллельная обработка данных. Издательский центр «Академия», Москва, 2010г. ISBN 978-5-7695-5951-8 – 336с.
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. Труды Всероссийской суперкомпьютерной конференции «Научный сервис в сети Интернет», сентябрь 2009г., Новороссийск – 1с.
4. Андреев С. С., Давыдов А. А., Дбар С. А., Лацис А. О., Плоткина Е. А. О разработке и испытаниях суперкомпьютеров с нетрадиционной архитектурой в ИПМ. Тезисы докладов 18-й Всероссийской конференции памяти К. И. Бабенко, Дюрсо, 2010г. – 1с.

5. Андреев С. С., Давыдов А. А., Дбар С. А., Лацис А. О. Плоткина Е. А. О моделях и технологиях программирования суперкомпьютеров с нетрадиционной архитектурой. Труды Международной суперкомпьютерной конференции «Научный сервис в сети Интернет», сентябрь 2010г., Новороссийск – 1с.
6. www.green500.org. Сетевой ресурс. Время доступа 11.10 4 октября 2011г.
7. http://www.kiam.ru/MVS/resourses/k100.html. Сетевой ресурс. Время доступа 11.25 4 октября 2011г.
8. http://www.kiam.ru/MVS/documents/k100/shmemprogman.html. Сетевой ресурс. Время доступа 11.30 4 октября 2011г.
9. www.qlogic.com. Сетевой ресурс. Время доступа 11.55 4 октября 2011г.
10. http://www.kiam.ru/MVS/documents/k100/userguide.html. Сетевой ресурс. Время доступа 12.50 4 октября 2011г.
11. http://www.netlib.org/scalapack. Сетевой ресурс. Время доступа 13.10 4 октября 2011г.
12. www.mcs.anl.gov/petsc Сетевой ресурс. Время доступа 13.10 4 октября 2011г.
13. http://www.intel.com/cd/software/products/emea/rus/379901.htm. Сетевой ресурс. Время доступа 14.55 4 октября 2011г.
14. http://fpga.parallel.ru/information.html. Сетевой ресурс. Время доступа 11.55 4 октября 2011г.
15. www.xilinx.com. Сетевой ресурс. Время доступа 11.55 4 октября 2011г.
16. http://www.kiam.ru/MVS/research/avtokod/index.html. Сетевой ресурс. Время доступа 11.55 4 октября 2011г.