Архитектура и программное обеспечение гибридных вычислительных систем на примере суперкомпьютера «МВС-Экспресс».

Лацис А. О.

Зав. сектором нетрадиционных суперкомпьютерных архитектур ИПМ им. М. В. Келдыша РАН, д.ф.-м.н.

lacis@kiam.ru

Москва, 2010

Прогноз на ближайшее будущее

- Экзафлопсные системы будут содержать миллионы процессорных ядер.
- Чтобы их строить, надо будет сначала договориться со знакомой электростанцией по соседству.
- И соорудить многоэтажный корпус с холодильниками.
- А потом решать проблемы ненадежности узлов, как и проблемы файловых систем с миллионами клиентов при единственном сервере.
- А заодно – проблемы глобального барьера по нескольким миллионам процессорных ядер.
- Все это, безусловно, будет, поскольку оно, по большому счету, уже есть.
- Но, может быть, есть и другой путь?
- Чтобы ответить на этот вопрос, мало знать, «что» и «как» в сегодняшних суперкомпьютерах. Важнее исследовать главный вопрос – «почему»?

Рекомендуемая литература.

1. http://parallel.ru
1. Параллельные вычисления.
   В. В. Воеводин, Вл. В. Воеводин. БХВ-Петербург, Санкт-Петербург, 2002.
1. http://www.kiam.ru
1. Параллельная обработка данных.
   А. О. Лацис, Изд. центр «Академия», М., 2010.

Как и почему суперкомпьютеры стали такими, каковы они сегодня?

- Это важно знать, чтобы понять, какими они станут завтра.
- Краткий исторический обзор.

  - появление первых компьютеров, осознание роли устройства управления фоннеймановского типа (1950г.),

  - развитие суперкомпьютеров по «почти фоннеймановскому» пути (1950-1990гг.),

  - системы массового параллелизма, кластеры (1990-2007гг.)

  - «почти фоннеймановский» путь практически исчерпан. Обострение проблемы «паровозного кпд» как следствие роста числа транзисторов на кристалле. Взрывной рост интереса к альтернативным архитектурам вычислителя (2007г.)
- Мы стоим на пороге глубинного пересмотра архитектуры суперкомпьютера в самом фундаментальном смысле этого слова, вплоть до разрушения стены между разработкой hard- и software (этой стеной был традиционный процессор, а он перестал нас устраивать).
- Вся иерархия суперкомпьютерных технологий, начиная с методов и вычислительных процедур и кончая железом, в самое ближайшее время будет меняться так радикально, как еще никогда не менялась в истории информатики. Изменения эти ***вынужденные***, они произойдут, как бы нам ни хотелось обратного.
- Фундаментальных трудностей всего две:

  - как такие машины строить,

  - как их потом использовать. Эта трудность – главная.

Фундаментальные трудности и фундаментальные ограничения.

- **Правило экономии сил разработчика**. Мы не можем позволить себе работать с нуля. Такую работу никто не оплатит. Или, что то же самое, не будет ждать ее результатов (время - деньги).

  В отличие от ситуации времен БЭСМ-6, компьютерный ширпотреб выпускается действительно массовыми тиражами, в то время как суперкомпьютеры были и всегда будут штучными изделиями. Прямой конкуренции с ширпотребом нам не выдержать.

  Значит, новые возможности надо искать, в конечном счете, на рынке. Не придумывать, а искать.
- **Принципиально новые архитектуры потребуют принципиально новых моделей программирования**. «Гораздо более других», нежели потребовал переход к параллельному программированию 20 лет назад.

  Как показал уже опыт перехода к параллельному программированию, традиционная модель программирования принципиально ориентирована именно на фоннеймановский процессор.

Высокопроизводительные вычислители нетрадиционной архитектуры: пространство решений.

	Монолитные	Гибридные
Программи-руемые	Шкафы Nvidia Tesla	МВС-Экспресс (ИПМ РАН, НИИ «Квант»)
Реконфигу-рируемые	Продукция НИИ МВС (Таганрог)	Скиф-Аврора Cray XT5G,…

Преимущества гибридных машин.

- Большая и в целом неподъемная задача – создание совершенно новых машин и новых технологий их использования – распадается на несколько гораздо более мелких и простых, которые можно решать независимо. Убедимся в этом.
- Преимущества гибридного подхода с точки зрения изготовления суперкомпьютеров вполне очевидны.
- Важнейший шаг в переносе приложений на новые машины – предварительная глубокая структурная перестройка алгоритма с целью обеспечения максимальной локальности обращений к памяти. Гибридная природа машины позволяет выполнить эту работу отдельно от освоения новых архитектур и моделей программирования, причем постепенно, за несколько шагов.
- Результатом является выделение очень небольших фрагментов текста с интенсивными вычислениями над небольшим объемом данных – ***вычислительных ядер***. Переписывать на новых языках надо будет только их. Ядра эти просты и компактны – для их реализации можно разрабатывать простые и компактные языки, не засоренные сложными техническими деталями (сравните трудоемкость как проектирования, так и освоения языка Бэйсик 25-летней давности и языка C++ в современном виде).
- Структура приложения на этом этапе инвариантна к конкретному типу ускорителя.

Итоговый список частных задач.

- Как мы видели на прошлом слайде, большую часть работы нам удалось выполнить, не привлекая понятий новых моделей программирования вообще (наш лозунг: **гибридной машине – гибридное программирование**).
- Тем не менее, меньшая часть никуда не делась, хотя и распалась на две независимых задачи:

1). Разработка и освоение языков «камерного жанра» для разных типов сопроцессоров – ускорителей,

2). Разработка усовершенствованной коммуникационной системы. Этот пункт требует пояснений. С него и начнем.

Требования к коммуникационной системе гибридного суперкомпьютера.

- Гибридная программа предъявляет очень высокие требования к системе коммуникаций, как по сложности логики, так и по интенсивности. Важна величина задержки на акт коммуникации.
- Очень желательна коммуникационная система, гораздо более простая и эффективная, чем принято в кластерах.
- Хотелось бы, чтобы программы было гораздо проще писать и отлаживать, чем обычно, но притом накладные расходы на коммуникации (во всех смыслах) были гораздо меньше, чем обычно.
- Откуда такую коммуникационную систему брать?

Замечание о соотношении коммуникационной аппаратуры и модели программирования.

- Просто искать «хорошую модель программирования» мы не можем – нам важно, чтобы она допускала эффективную реализацию (попробуйте реализовать прекрасную модель программирования OpenMP на грид-системе).
- Искать надо хорошее коммуникационное оборудование, отвечающее нашим требованиям, а для него – подходящую модель.
- К счастью, те же процессы эволюции электронного оборудования, которые вызвали проблему «паровозного кпд», одновременно породили ряд очень полезных новых возможностей в области коммуникаций.

Еще одно следствие роста числа транзисторов на кристалле.

- Неудержимый рост числа транзисторов на кристалле имел, как минимум, одно положительное следствие.
- Стал возможным (и произошел) переход от шинного способа интеграции устройств в рамках одного компьютера к сетевому (Hypertransport, PCI Express).
- В результате не только повысилась «коммуникационная вооруженность» узла в кластере, но и начала размываться грань узла как такового.
- Магистрали PCI Express (в отличие от старого, шинного варианта PCI) сравнительно легко соединять друг с другом в рамках всего кластера, получая единую NUMA-систему. Чужая память видна узлу примерно так же, как память собственной видеокарты.
- Получается NUMA-система с высокой степенью асимметрии доступа, к сожалению, не кэш-когерентная (аппаратура обеспечивает только буферизованную попарно когерентную запись в чужую память, но чтение чужой памяти кэшировать нельзя).
- Многократно снижаются задержки запуска единичного обмена (для записи – в десятки раз по сравнению с сетями класса Infiniband).

Как программировать для таких систем?

- Интуитивно очевидно, что MPI – слишком слабая модель для такого оборудования, не способная в полной мере донести до программиста его возможности.
- OpenMP тоже не годится – принципиально отсутствие кэш-когерентности и высокая степень асимметрии доступа.
- Нужно что-то среднее. Что именно?
- Ответ дает практика использования продукции Cray и SGI. Машины этих производителей уже многие годы оснащаются коммуникационными системами со сходными свойствами, просто раньше (до появления PCI Express) такие коммуникационные системы очень дорого стоили. Но те, кто мог себе их позволить, накопили определенный опыт, и сформировали свою базовую технологию параллельного программирования.
- Называется она ***shmem*** (от ***sh***ared ***mem***ory), и представляет собой библиотеку односторонних обменов, ориентированную на низкие задержки.
- Или, что то же самое, библиотеку доступа к единой разделяемой памяти при помощи функций доступа.
- Пример функции: «Значение локальной переменной A присвоить той переменной B, которая находится в процессе N».
- Синхронизация – барьеры, не обязательно глобальные, и атомарные операции.
- Коллективные операции также предусмотрены.
- Такую модель естественно рассматривать как простейший, низкоуровневый вариант PGAS. Из более высокоуровневых ей в наибольшей степени соответствует Co-Array Fortran.

Коммуникационная система: основные выводы.

- Современному состоянию технологий, применяемых в компьютерном ширпотребе, соответствует новый класс коммуникационных систем, промежуточный между привычными нам классами SMP и MPP.
- Этому оборудованию хорошо отвечают технологии программирования класса PGAS (не зря они в последнее время так популярны).
- Такие коммуникационные системы хороши сами по себе, но особую ценность приобретают в составе гибридных машин.
- По своим потребительским свойствам эти технологии ближе к технологиям работы с общей памятью, чем к MPI. В частности, они допускают пошаговое распараллеливание, хорошо сочетаются как с OpenMP, так и со средствами программирования сопроцессоров-ускорителей при использовании многоуровневого параллелизма.

Откуда брать такое оборудование?

- Источников три.

1). Можно изготовить его на основе серийно выпускаемых массовыми тиражами комплектующих. Так сделано в МВС-Экспресс.

2). Можно реализовать в программируемой логике. Так сделано в Скиф-Авроре. Путь этот не дешевый, но, как мы увидим дальше, многообещающий.

3). Можно попытаться реализовать эту модель программирования в современных версиях Infiniband. Вопрос о том, велики ли будут потери, пока представляется открытым, хотя уже интенсивно исследуется.

- Мы отделили и рассмотрели по отдельности столько частных проблем создания машин с новой архитектурой, сколько смогли. Теперь рассмотрим собственно проблему программирования для новых архитектур. В нашем случае она приобрела форму проблемы программирования вычислительных ядер для сопроцессоров.

Вычислительные ядра на программируемых процессорах.

- Для ускорителей на базе видеокарт имеются прекрасные технологии программирования, легко осваиваемые и дающие многократное ускорение по сравнению с универсальными процессорами.
- По сравнению с универсальными процессорами, в расчете на ватт или на транзистор, достигается впечатляющее быстродействие.
- НО!!! В расчете на собственный заявленный пик, получаем в точности тот самый «паровозный кпд».
- Мы взяли то, что легко было взять. Можно ли взять еще больше?
- Можно, но только ценой смены парадигмы. Границу между «soft» и «hard» надо научиться проводить не между программой и процессором, а между схемотехническим и радиоэлектронным уровнями представления схемы. Надо научиться строить прикладные схемы – процессоры одной задачи, причем так же легко, как мы сегодня пишем программы.

Технологии разработки схем на ПЛИС. Основные факты.

- ***Интегральная схема программируемой логики*** (ПЛИС) – универсальная заготовка, способная превратиться в любое цифровое электронное устройство, если в нее загрузить описание схемы этого устройства. Выпускаются массовыми тиражами.
- Для наиболее мощных ПЛИС – ***FPGA*** – загрузка может выполняться неограниченное число раз, и занимает от секунд до десятков секунд.
- Современные FPGA содержат все необходимое для реализации высокопроизводительных каналов связи, совместимых с PCI Express.
- Проектирование описания требуемой схемы выполняется в логических терминах. Корректность схемы в терминах радиоэлектроники обеспечивается автоматически. В отличие от старинных технологий проектирования цифровых схем, схемотехник не обязан знать электронику.
- FPGA на порядок медленнее по рабочей частоте, чем «жесткая» логика программируемых процессоров.

Вычислительная схемотехника на FPGA. Основные факты.

- Идея использования реконфигурируемых сопроцессоров в машинах общего назначения старше, чем GPGPU. Такие машины выпускали Cray и SGI.
- Реализуемость и эффективность конкретных «процессоров одной задачи», в том числе – для интенсивных вычислений с двойной точностью, неоднократно доказана экспериментально.
- Главной проблемой считается длительность и трудоемкость разработки, необходимость специальной схемотехнической подготовки.
- Технологии прикладного схемотехнического проектирования, ориентированные на конечного пользователя, бурно развиваются в последние годы, но хороших систем пока нет. Потери по сравнению с грамотной профессиональной реализацией – почти порядок.

А вот теперь – требуется полет фантазии.

- Есть актуальная проблема. Решив ее, мы сможем делать с оборудованием **все**. «Холодные» схемы с почти стопроцентным кпд, скромно размещаемые в уголке материнской платы и обгоняющие все ее процессоры вместе в разы, станут повседневной реальностью.
- Есть абсолютно все технические предпосылки решения проблемы.
- Нет решения. Не потому, что чего-то не хватает, а потому, что … **не придумано**.
- Это надо осмыслить. Такое бывает не каждые 10 лет.
- Такие ситуации в технике принято называть «окном прорывных возможностей, которое скоро закроется». Без нас придумают. Обидно.
- Тем более обидно, что не требуется даже фундаментальной переподготовки. Программирование – логика, и схемотехника – логика. Системные программисты всегда считали себя великими логиками, любителями и творцами новых, красивых формализмов.
- Ну и?

Образмерим проблему.

- Технологии программирования, применяемые для GPGPU, Cell и им подобных, не годятся. На схемном уровне совершенно другой вид параллелизма – глубокий конвейер, и другой стиль проектирования – скорее структурный, чем алгоритмический. Попытки трансляции, скажем, CUDA в схему закономерно дают плохой результат.
- Технологии программирования, заведомо пригодные для разработки эффективных схем, существуют (VHDL, Verilog), но считаются совершенно непригодными для прикладных программистов (и вполне справедливо).
- Разрабатываемые в качестве альтернативы технологии оказываются равно не пригодными и для прикладного программиста, и для оборудования. В чем дело?
- Похоже, что в нашей умственной лени и программистском снобизме.

Трудная судьба схемотехнической программистской модели.

- VHDL и Verilog разрабатывались схемотехниками для себя. Авторы не ставили цели прояснить программистскую модель через язык – у схемотехника модель мира к моменту изучения VHDL уже сформирована другими способами.
- Языки эти традиционно развивались как «три в одном» – как языки спецификации, моделирования и синтеза схем одновременно, а это – совсем разные вещи.
- В результате получились языки, не соответствующие собственной модели. Изучив Фортран, Вы поняли, что такое фоннеймановская машина. Изучив VHDL, Вы поняли только, что все очень сложно, и надо, видимо, «знать электронику» (что неправда).
- Схемотехническая модель программирования существует, но в терминах специалистов по программистским моделям не имеет даже названия. Нет хорошего языка, по которому ее бы можно было изучить. «Знать электронику» надо только для того, чтобы пробиться через скверное качество существующих языков!
- Так может быть, нужно ее (эту модель) просто «правильно приготовить»? Придумать в ее рамках нормальный язык?
- Уже одно это дало бы как громадное продвижение в схемотехническом проектировании вычислительных ядер, так и очень много в понимании устройства более высокоуровневых инструментов такого проектирования. Мы работаем в этом (и не только) направлении.

Еще раз об окне возможностей.

- Конкретный рассказ о схемотехнической модели программирования – тема отдельной лекции.
- Отдельно отметим, что совместное использование FPGA для реализации как коммуникационной системы, так и вычислительных ядер, видимо, способно дать заметный синергетический эффект, привести к обобщению гибридного подхода.
- Уже сказанного здесь достаточно, чтобы утверждать, что схемотехническое проектирование как вид совместной деятельности системных и прикладных программистов в области высокопроизводительных вычислений – вещь крайне актуальная и перспективная.
- Бум технологий GPGPU временно отодвинул эту тематику в тень, дав нам еще один шанс крепко подумать и, быть может, успеть вскочить в последний вагон этого поезда.
- Однако же, расслабляться не следует. Потом будем жалеть.

Спасибо за внимание.

Вопросы?