Праці міжн. симпозіума "Комп"ютери у Європі. Минуле, сучасне та майбутнє". –Київ: ІК. - 1998р. -С.460-467.

ЛОКАЛЬНО СВЯЗАННЫЕ ВЫЧИСЛЕНИЯ - СОСТОЯНИЕ И ПЕРСПЕКТИВЫ

Каневский Ю.С., Сергиенко А.М.
Национальный технический университет Украины “КПИ”, г. Киев, Украина.

Введение.

В настоящее время параллельные вычислительные системы (ВС) с массовым параллелизмом являются практически незаменимыми при решении крупномасштабных исследовательских и коммерческих задач в физике, химии, биологии, электронике, конструировании, медицине, экономике и других отраслях.

При разработке параллельных вычислительных приложений наибольшее распространение получила модель ВС с разделяемой памятью. Она проста для понимания и программирования, но не обладает свойством масштабируемости. Поэтому вычисления с массовым параллелизмом на ней невозможны. Это связано с тем, что если общие переменные хранятся в разделяемой памяти, то физически невозможно реализовать к ним достаточно быстрый бесконфликтный доступ в заданном порядке от большого количества процессорных элементов ВС.

Часто используют модель параллельной ВС, процессорные узлы которой представляются точками в многомерном пространстве. Если при выполнении алгоритма передачи данных осуществляются, в основном, между соседними процессорными узлами, (расстояние между которыми в n-мерном пространстве равно единице), то такие вычисления называют локально связанными.

Локально связанные вычисления характерны, например, для задач математической физики, вообще для задач, решаемых методами линейной алгебры, задач обработки сигналов и изображений и др. Алгоритмы решения этих задач являются, как правило, локально-рекурсивными. Их реализация за приемлемое время требует применения систем с призводительностью порядка 10^{10}-10^{20} операций в секунду [1], что интенсифицировало разработку как методов распараллеливания алгоритмов и программ, так и методов проектирования высокопараллельных систем. В связи с изложенным в области локально связанных вычислений получены, пожалуй, самые серьезные результаты по автоматическому распараллеливанию алгоритмов и их отображению в масштабируемые системы с массовым параллелизмом. Вместе с тем, наряду с большими достижениями и перспективами, в этой области существует и множество нерешенных проблем, которые будут рассмотрены ниже.

Аппаратное обеспечение.

Широко известными примерами локально связанных структур являются структуры с графами типа гиперкуб, решетка, дерево. К ним относятся также кольцевые, тороидальные, а также комбинированные структуры, такие как пирамида, решетка кластеров и другие. Среди локально связанных структур решетчатые структуры отличаются масштабируемостью и сравнительно невысокими аппаратурными и временными затратами на организацию межпроцессорных обменов. Масштабируемость означает, что наращивание количества процессорных элементов (ПЭ) системы влечет за собой линейный рост вычислительной мощности. При этом это наращивание выполняется единообразно и локальность связей в такой системе нередко оказывается не только топологической, но и физической. Одно- и двумерная решетки наиболее эффективны при реализации в СБИС. Большой класс алгоритмов обработки сигналов, линейной алгебры и других находит эффективное отображение именно в решетчатые структуры.

В свое время в нашей стране широко использовалась вычислительная система ПС2000 с линейной и тороидальной структурой. Многие коммерческие параллельные ВС, такие как Intel Paragon, Cray T3-D, также имеют структуру решетки. Однако, высокая цена высокопроизводительных вычислителей является существенным барьером для их широкого распространения.

Разработка и производство сравнительно дешевых высокорпоизводительных микропроцессоров позволяют существенно снизить стоимость параллельных ВС. В этой связи при анализе тенденций развития архитектур параллельных ВС следует принять во внимание следующие доводы и закономерности.

- С периодом 1,5 - 2 года удваивается сложность и быстродействие СБИС и ученые и технологи не видят барьеров для этого роста в дальнейшем. Однако, с ростом степени интеграции, начиная с определенной границы проектных норм кристалла, нарушается широко используемая масштабируемость топологии. При этом в то время как постоянно повышается быстродействие вентилей, удельная линейная проводимость металлических проводников уменьшается быстрее, чем их емкость и в результате растут задержки распространения в них сигналов. Это приводит к тому, что для передачи сигналов в пределах кристалла необходимо обеспечивать их буферизацию и запоминание в регистрах через короткие линейные промежутки. Так, при проектных нормах 0,06 мкм, которые предполагается освоить к 2010 году, расстояние между смежными буферами и регистрами должно быть не более 0,12 мм и 1,5 мм, соответственно. В результате, связанные вычисления и управляющие операции, выполняемые в течение одного такта, могут охватывать всего лишь порядка 2 млн. из 400 млн. вентилей на кристалле [2]. Уже в существующих кристаллах более 80% их поверхности занимают линии связи, на которых рассеивается 80% энергии переключения. Это означает, что проектирование новых архитектур должно вестись с соблюдением принципа близкодействия, который для локально связанных структур является естественным.

- В настоящее время широкое распространение получили однокристальные суперскалярные микропроцессоры, которые будут играть ведущую роль и в начале следующего столетия. Архитектурные совершенствования суперскалярного микропроцессора за счет увеличения аппаратных затрат в десятки и сотни раз приведут к повышению его производительности всего лишь в 4-9 раз [3]. Вместе с тем при размещении нескольких десятков и сотен локально связанных ПЭ на кристалле общая производительность системы для соответствующих классов алгоритмов может расти линейно.
- Процесс проектирования работоспособных кристаллов микропроцессоров постоянно усложняется и дорожает. Так, фирма Intel тратит на верификацию и тестирование проектов 40-50% всех расходов на проектирование. С этой точки зрения не емкость кристалла, а сложность реализуемой функциональной схемы, умноженная на коммерческий интерес фирмы-изготовителя устанавливает верхнюю границу сложности проектируемой микросхемы [4]. Естественно, что регулярную ВС с локальными связями гораздо проще проектировать и тестировать, чем единичный процессор с таким же объемом оборудования.
- Бурное развитие систем телекоммуникации и мультимедиа привело к введению архитектурной поддержки в суперскалярные архитектуры в виде расширения системы команд MMX (Intel), MVI (Digital), MAXZ (Hewlett Packard) и др. Рынок приложений мультимедиа растет и будет динамично развиваться. Эти приложения характерны тем, что они требуют постоянно растущего объема вычислений, которые в большинстве случаев являются локально связанными. С точки зрения программиста, аппаратную часть, реализующую команды MMX, можно рассматривать как четырех- или восьмипроцессорную линейную ВС. Вполне закономерно, что рост аппаратных затрат на архитектурную поддержку приложений мультимедиа в дальнейшем будет опережать рост этих затрат на модернизацию суперскалярного ядра процессоров [5].

Из вышесказанного можно сделать вывод о необходимости массового выпуска в течение ближайшего десятилетия кристаллов, содержащих десятки, может быть, сотни локально связанных ПЭ. В работе [6] уже предложен проект кристалла процессорной матрицы со 128 ПЭ, каждый из которых имеет сложность микропроцессора Alpha 21064. Массовое производство подобных кристаллов обеспечит широкое внедрение локально связанных систем не только в таких областях, как обработка сигналов, телекоммуникация, мультимедиа, но и многих новых, например, в персональных супер-ЭВМ. Сейчас считается, что ВС с высокой степенью параллелизма содержит 512 и более ПЭ [7]. При использовании новых многопроцессорных кристаллов коммерческие ВС с массовым параллелизмом, предназначенные для реализации определенных классов алгоритмов, могут содержать десятки и сотни тысяч ПЭ.

Программное и алгоритмическое обеспечение.

С появлением параллельных, в том числе матричных ВС возникла проблема создания для них матобеспечения. Разработка параллельной

программы, как правило, представляет собой сложный ручной, высококвалифицированный и непроизводительный труд. Дороговизна параллельного матобеспечения определяется также тем, что оно непереносимо между различными архитектурами и тем, что круг пользователей очень ограничен. Поэтому постоянно велись работы по созданию автоматизированных средств для программирования этих ВС.

Программные циклы считаются основными источником массового параллелизма в пользовательских программах. Их отображению в ВС с локальными связями посвящено большое количество работ, результаты которых нашли применение в распараллеливающих компиляторах. Так, метод гиперплоскостей [8] нашел применение в компиляторе для решетки процессоров IlliacIV и послужил основой для целого направления аналитических методов распараллеливания циклов. Широко известны методы распараллеливания циклов, разработанные нашими соотечественниками, например, [9,10,11]. Однако, предыдущие десятилетия ознаменовались периодом расцвета конвейерных супер-ЭВМ и основная масса сил была направлена на создание матобеспечения для этого класса ВС.

В результате, на конференции по параллельным вычислениям ICCP'97 [7], посвященной двадцатипятилетлетней истории этой отрасли науки и техники, отмечалось, что существующее состояние параллельного матобеспечения для ВС с массовым параллелизмом следует признать далеким от желаемого. При существующей технологии создания параллельных программ затраты на его разработку никогда не окупаются. Причинами этого являются: 1) неудовлетворительная работа распараллеливающих компиляторов; 2) то, что эффективные программы приходится писать на языке ассемблера с учетом всех архитектурных особенностей ВС; 3) отсутствие переносимого матобеспечения, включая и библиотеки процедур; 4) низкая эффективность применяемых программных моделей, таких как MPI, PVM; 5) отсутствие эффективных отладочных и вспомогательных средств. Кроме того, обычно пользователь считает, что лучше обращаться с персональным единичным, хотя и значительно более медленным, зато удобным для понимания процессором, чем с мощной разделяемой многопроцессорной ВС. И наконец, слишком большая дистанция между разработчиками таких ВС и конечными пользователями не способствует их эффективному использованию [7].

С каждым днем появляются новые приложения, требующие больших объемов вычислений. Это решение таких задач, как распознавание образов, построение виртуальной реальности, моделирование различных объектов и веществ, анализ рынка и т.п., которые нередко должны решаться в реальном масштабе времени. Этот процесс заставляет постоянно наращивать вычислительные мощности и требует разработки более сложного параллельного матобеспечения. Для того, чтобы параллельные, в том числе матричные ВС и процессорные матрицы получили массовое распространение, необходимо решить ряд следующих задач, касающихся матобеспечения.

- Построение новых эффективных языков параллельного программирования. Во-первых, язык должен в большей степени отражать особенности предметной области, в которой решается задача, во-вторых, его

конструкции должны иметь эффективное отображение в архитектуру целевой ВС, в-третьих, он должен быть понятным и удобным в пользовании, как например, язык визуального программирования.

- Разработка эффективных распараллеливающих компиляторов, которые, с одной стороны, позволили бы обычному пользователю работать с реальной параллельной ВС как с привычной и понятной программной моделью, с другой стороны, обеспечили бы высокую загруженность этой ВС. Низкая эффективность существующих компиляторов во многом объясняется тем, что они создавались под заданную архитектуру ВС, в настоящее время намечается тенденция разработки архитектуры под эффективный оптимизирующий компилятор. В этой связи необходимо создание новых программных моделей ВС, которые были бы и понятны пользователю, и имели эффективное, в том числе масштабируемое отображение в физическую модель ВС.

Примером такой модели может служить ВС с виртуальной топологией. Применяя такую модель, программист составляет параллельную программу для удобной ему и эффективной для задачи (виртуальной) структуры ВС, однако и с такой моделью за программистом остается решение сложной задачи отображения алгоритма в архитектуру ВС.

- Разработка новых параллельных алгоритмов. Опыт трансляции последовательных программ в параллельные программы, а следовательно, и алгоритмов, созданных для монопроцессорной системы, показал крайне низкую эффективность такого подхода к созданию параллельного матобеспечения [7,11].

До сих пор нередко считается, что лучший вычислительный алгоритм тот, который имеет минимум арифметических операций, особенно трудоемких. Однако для параллельных вычислений важным является также учет сложности межоператорных связей по данным и управлению. Так, алгоритм Левинсона решения системы уравнений с теплицевыми матрицами имеет на порядок меньшее число операций, чем, например, алгоритм Гаусса. Но по причине сложности межоператорных связей его выполнение, например, на процессорной матрице может занять больше времени [12]. Поэтому нобходима разработка новых алгоритмов, которые являются результатом “параллельного“ мышления и которые имеют эффективное отображение в параллельные архитектуры. Паралельную версию алгоритма можно получать, например, путем построения графа алгоритма с однократными присваиваниями и дальнейшего его гомоморфного преобразования в адекватные алгоритму вычислительные схемы (исполнительные алгоритмы).

- Разработка переносимых библиотек процедур и ядер. Переносимое параллельное матобеспечение - это ключевой фактор окупаемости разработки матобеспечения. Необходимо также проектирование отладочных и вспомагательных программных средств, таких как средства ввода-вывода, обслуживания контрольных точек и перезапуска задачи и т.п.

Вычисления на уровне операторов и макроопераций.

Конвейерная обработка команд и данных является самым распространенным видом параллельных вычислений, поскольку она

присутствует во всех современных суперскалярных микропроцессорах. Рассматривая конвейер как линейную структуру, каждый модуль которой выполняет свою функцию, вычислительный процесс в нем можно также считать локально связанным. В суперскалярных микропроцессорах исполненяемые команды проходят от пяти до четырнадцати и более ступеней конвейеров в каждом из нескольких обрабатывающих блоков (потоков), причем с ростом степени интеграции число ступеней конвейера растет. Ожидаемое ускорение вычислений в таких процессорах достигается за счет динамической диспетчеризации выполнения команд в обрабатывающих блоках. Однако оно, как правило, далеко от предельного вследствие частой перезагрузки конвейеров из-за взаимной несогласованности команд и данных в исходном потоке команд и данных. Поэтому аппаратно реализованные механизмы этой диспетчеризации постоянно усложняются. Для этого в новых процессорах, наряду с такими блоками, как блок предсказания перехода, буферы готовых к обработке данных, делается ставка на использование громоздких ОЗУ трассы, блоков предсказания данных и результатов [3]. Однако, накладные расходы на организацию вычислений, связанные с динамическим устранением конфликтов по данным, управлению и ресурсам, все равно остаются большими.

В связи с этим для минимизации этих расходов предполагается, переложить часть функций по планированию вычислений с аппаратуры процессора на компилятор, как например, в проекте Merced. Это приведет к ускорению вычислений, однако, значительным оно не будет в силу того, что распараллеливание и планирование вычислений выполняется все еще на операторном (арифметическом) уровне, а не на алгоритмическом.

В последнее время получают широкое развитие программируемые логические интегральные схемы (ПЛИС или FPGA) как вычислительная среда для аппаратной реализации некоторых функций и алгоритмов с большим числом нестандартных операций, например, специфической битовой обработки. Во многих случаях такая реализация параллельных алгоритмов имеет стоимостное и временное преимущество над их реализацией на многопроцессорных ВС [6]. К концу столетия ожидается применение микросхем FPGA с емкостью до одного миллиона элементарных логических модулей. Однако, при применении FPGA возникают следующие проблемы, связанные с организацией в них высокопроизводительных вычислений.

Трассировка схемы со сложными нелокальными межмодульными связями занимает десятки часов машинного времени и нередко заканчивается малоэффективным решением. Начиная с частот порядка 100-200 МГц, дальнейший подьем рабочей тактовой частоты FPGA невозможен, если связи между запрограммированными функциональными модулями нелокальные [6].

Поэтому вышеперечисленные задачи разработки матобеспечения для локально связанных вычислений тесно взаимосвязаны как с повышением загруженности вычислительных блоков суперскалярных микропроцессоров, так и с повышением эффективности использования FPGA.

Процессорная матрица как модель локально связанных вычислений.

Процессорная матрица представляет собой одно- или двумерную решетку как правило однотипных ПЭ. Полностью локально связанные вычисления в этой матрице организованы таким образом, что исходные данные синхронно загружаются, а результаты выгружаются через граничные ПЭ, причем этот процесс выполняется на фоне вычислений. Масштабирование этой структуры при увеличении размеров задачи или необходимости сокращения времени ее решения осуществляется единообразным подсоединением ПЭ к краям матрицы.

Хронологически первыми были систолические матрицы. Они появились в начале прошлого десятилетия как структуры специализированных СБИС для высокопроизводительной обработки данных, но по ряду причин широкого практического применения не нашли. Тем не менее фактически они представляли собой одну из первых моделей ВС с массовым параллелизмом. При разработке методов их проектирования и создании систолических алгоритмов получены серьезные результаты, основными из которых являются следующие.

- Систолическую матрицу рассматривают как аппаратную форму задания параллельного алгоритма. От такой формы задания алгоритма несложно осуществить формальный переход к параллельной программе для ВС с заданными архитектурой и числом ПЭ, то есть можно в большой мере обеспечить переносимость матобеспечения. Причем такая программа позволяет получить высокую загруженность ПЭ, масштабируемость задачи и архитектуры, минимум используемой оперативной памяти и межпроцессорных обменов, которые являются локальными. Весьма полезным эффектом такого отображения является также минимизация конфликтов кеш-памяти реальных ПЭ, поскольку обмены данными заранее спланированны и выполняются на фоне вычислений. Поэтому был введен специальный термин “систолическое программирование”, который означает разработку параллельной программы реализации машинного алгоритма, заданного в виде систолической структуры.
- Разработаны формальные методы отображения регулярных алгоритмов, задаваемых гнездом циклов, в систолические матрицы. Как и метод гиперплоскостей [8], многие из этих методов основаны на отображении пространства итераций алгоритма в ортогональные подпространства структур и событий. Однако существенным их отличием является то, что отображению в структуру вычислителя подвергаются не только операторы, но и зависимости по данным, что позволяет целенаправленно искать локально связанные структуры [13,14].
- Разработаны различные методы и способы преобразования алгоритмов со сложными межитерационными связями в регулярные алгоритмы. Применяются также различные способы сокращения числа или исключения логических операторов. В сочетании с методами синтеза систолических матриц они позволяют отображать большой класс алгоритмов в ВС с локальными связями.

- Разработаны методы иерархического отображения составных гнезд циклов со сложным ядром. Некоторые из них учитывают конвейерный характер выполнения операторов в процессорах с суперскалярной архитектурой [15].
- С помощью новых методов разработаны систолические версии большого круга параллельных алгоритмов решения задач обработки сигналов и изображений, линейной алгебры и многих других. Число таких алгоритмов и их структурная сложность постоянно увеличиваются.

Локально связанные структуры могут служить эффективной программной моделью ВС, а методы их синтеза и преобразования - основой для создания соответствующего распараллеливающего компилятора. Такой компилятор будет обеспечивать локальность вычислений в параллельной ВС при выполнении скомпилированной программы на заданном количестве ПЭ и соответственно, реализацию всех преимуществ локально связанных вычислений и, наконец, переносимость исходной программы между различными моделями ВС с обеспечением локальных обменов между ПЭ.

Заключение.

Постоянный рост объемов вычислений при решении крупномасштабных исследовательских и коммерческих задач, а также степени интеграции микросхем должны привести к большему распространению вычислений с массовым параллелизмом на вычислительных системах, состоящих из многих сотен и тысяч процессоров и расширить сферу использования высокопроизводительных вычислений, включая настольные суперкомпьютеры.

Однако в настоящее время мало потребителей и программистов в странах мира используют параллельную вычислительную технику. Основной из причин этого является отсутствие желания заниматься трудоемкой и малопонятной работой по разработке параллельных алгоритмов и программ. Это связано также с отсутствием эффективных языков параллельного программирования, компиляторов, средств отладки и поддержки параллельных задач, переносимости матобеспечения. В ближайшем будущем спрос на высокопроизводительные вычисления с аппаратной стороны с лихвой будет покрываться коммерческими моделями параллельных компьютеров. Но если не изменится существующее отношение к разработке соответствующего алгоритмического и программного обеспечения, то будет остро ощущаться в нем дефицит.

Применение локально связанных структур в качестве модели параллельных вычислений, а методов их синтеза и преобразования как базового аппарата для построения распараллеливающего, оптимизирующего компилятора поможет создать предпосылки для автоматизированной разработки программ решения большого круга задач на ВС с массовым параллелизмом.

1. Петерсон В.Л., Ким Дж., Холст Г.Л., Диверт Дж.С. и др. Требования к супер-ЭВМ в избранных дисциплинах, представляющих интерес для аэрокосмонавтики// ТИИЭР, -Т.77. -1989. -N7.-С.66-84.
2. Mazke D. Will Physical Scalability Sabotage Performance Gains? // Computer. -V.30. -1997. -N9. -P.37-39.

3. Lipasti M.H., Shen J.P. Superspeculative Microarchitecture for beyond ad 2000.// Computer. -V.30. -1997. -N9. -P.63-72.
4. Murray B.T., Hayes J.P. Testing ICs: Getting to the Core of the Problem. // Computer. -V.29. -1996. -N11. -P.32-38.
5. Diefendorf K., Dubey P.K. How Multimedia Workloads Will Change Processor Design. // Computer. -V.30. -1997. -N9. -P.43-45.
6. Waingold E., Taylor M., Srikrishna P., et al. Baring it all to Software: Raw Machines. // Computer. -V.30. -1997. -N9. -P.86-93.
7. Theys M.D., Braun T.D., Siegel H.J. Widespread Acceptance of General-Purpose, Large-Scale Parallel Machines: Fact, Future or Fantasy? //IEEE Concurrency. -V6. -1998. -N1. -P.79-83.
8. Lamport L. The Parallel Execution of DO Loops // CACM. -V17. -1974. - N2. -P.83-93.
9. Годлевский А.Б., Крат С.П. Об одном подходе к реализации динамического распараллеливания программ на многопроцессорных системах // Кибернетика. -1984. -№3. -С.114-117.
10. Капитонова Ю.В., Летичевский А.А., Бублик В.В., Коляда С.В. Об алгоритме максимальной десеквенции бесповоротных циклических операторов // Кибернетика. -1983. -№4. -С.11-17.
11. Вальковский В.В. Распараллеливание алгоритмов и программ. Структурный подход. - М.: Радио и связь, 1989. -176с.
12. Сверхбольшие интегральные схемы и современная обработка сигналов/ Под ред. С.Гуна, Х.Уайтхауса, Т.Кайлата. -М.: Мир. -1989. -472с.
13. Гун С. Матричные процессоры на СБИС. -М.: Мир, -1991. -248с.
14. Каневский Ю.С. Систолические процессоры.-Киев: Тэхника. 1991.-173 с.
15. Kanevski, J.S., Sergyienko, A.M., Piech H. A method for the structural synthesis of pipelined array processors // 1-st Int. Conf. "Parallel Processing and Applied Mathematics", PPAM'94, Czestochowa (Poland), Sept. 14-16, 1994. - P. 100-109.