РАСПРЕДЕЛЕННЫЕ ВЫЧИСЛЕНИЯ И МУЛЬТИПРОЦЕССОРЫ С ДИНАМИЧЕСКОЙ АРХИТЕКТУРОЙ

В. А. ТОРГАШЕВ

Санкт-Петербургский институт информатики и автоматизации РАН

СПИИРАН, 14-я линия ВО, д. 39, Санкт-Петербург, 199178;

<tor@spiiras.nw.ru>

УДК 681.3

Торгашев В. А. **Распределенные вычисления и мультипроцессоры с динамической архитектурой** // Труды СПИИРАН. Вып. 4. — СПб.: Наука, 2007.

Аннотация. *Рассмотрены проблемы организации распределенных вычислений и показано, что главной причиной возникающих трудностей является отсутствие адекватной вычислительной модели. Предлагается в качестве такой модели использовать динамические автоматные сети (ДАС). Рассматриваются свойства ДАС и возможности реализации мультипроцессоров с динамической архитектурой на базе этой модели. Показана ретроспектива развития и реализации данной концепции от рекурсивных вычислительных машин до мультипроцессоров с динамической архитектурой* — Библ. 21 назв.

UDC 681.3

Torgashev V. A. **Distributed Computing and Dynamic Architecture Multiprocessors** // SPIIRAS Proceedings. Issue 4. — SPb.: Nauka, 2007.

Abstract. *The problems of organizing distributed computations are considered. It is shown that the main reason for arising difficulties is an absence of adequate computational model. Dynamic automata networks (DAN) are proposed as such a model. Properties of DAN and possibilities of realizing multiprocessors with dynamic architecture based on this model are considered. Retrospective review of developing and realizing this concept from recursive computers to multiprocessors with dynamic architecture is shown.* — Bibl. 21 items.

1. Введение

Проблема распределенных вычислений, то есть вычислений, выполняемых совместно множеством вычислительных устройств, является одной из наиболее важных в вычислительной технике уже не одно десятилетие. Основными областями применения распределенных вычислений являются: супер-ЭВМ, информационно-вычислительные сети, недорогие ЭВМ класса мини-супер и высоконадежные ЭВМ, работающие в реальном времени. Несмотря на определенные успехи, достигнутые в данном направлении, кардинального решения проблемы нет до сих пор.

Действительно, в настоящее время существует множество супер-ЭВМ, таких например как «Blue Gene» фирмы IBM, состоящих из сотен тысяч вычислительных устройств на базе мощных микропроцессоров и позволяющих достичь производительности порядка сотен триллионов операций в секунду (сотни терафлопс). Однако для многих реальных задач не удается обеспечить эффективную загрузку большинства устройств таких супер-ЭВМ. Работы по использованию для распределенных вычислений корпоративных сетей (grid-технологии) или глобальной сети Интернет (метакомпьютинг) ведутся уже достаточно давно. Детально проработана методология, созданы бесплатно распространяемые программные продукты, включающие и исходные тексты (например, Globus Toolkit). Однако широкого распространения эти технологии до сих пор не полу-

чили. Более того, становится очевидным, что существующие подходы к организации распределенных вычислений не в состоянии решить основную задачу указанных работ — дать возможность *любому* пользователю использовать громадные вычислительные ресурсы существующих сетей, как локальных, так и глобальных, для решения собственных задач, так же свободно, как сейчас он использует информационные ресурсы этих сетей.

Существует несколько причин такого положения дел. Первая и главная причина заключается в том, что вычислительные ресурсы сетей в основном распределены по персональным компьютерам, принадлежащим конкретным индивидуумам на правах личной или корпоративной собственности. Указанные индивидуумы, в большинстве своем, не заинтересованы в распределенных вычислениях и не имеют реальной мотивации по предоставлению ресурсов своих ПК для общего пользования.

Помимо отсутствия мотивации существуют вполне обоснованные опасения, что чужие программы, выполняемые в ПК, могут привести к различным нарушениям работы ПК, уничтожению важных для пользователя данных или даже применяться в качестве вирусных или шпионских программ.

Наконец, безусловно слабым местом всех современных методов распределенных вычислений, в том числе и в суперкомпьютерах, является централизованный характер администрирования, включая распределение ресурсов (как правило, статическое), инициация задачи и контроль за ее прохождением в сети, контроль и диагностика программных и аппаратных ошибок. Все эти функции возлагаются обычно на один сервер или хост-процессор, что существенно ограничивает число одновременно выполняемых задач. Поэтому задача предоставления любому пользователю вычислительных ресурсов сетей для распределенных вычислений в практической плоскости даже не ставится.

В области малых недорогих вычислительных систем возможности распределенных вычислений практически не используются. За персональными ЭВМ, являющимися сугубо однопроцессорными, следуют рабочие станции и серверы, превосходящие ПЭВМ по цене в десятки раз, а по производительности — лишь в несколько раз. В то же время на рынке практически отсутствуют мультипроцессорные ПЭВМ, содержащие десяток или большее число микропроцессоров, которые по цене превосходили бы обычные ПЭВМ в 2–3 раза, а по производительности по меньшей мере в 10 раз, хотя технические возможности для создания таких ПЭВМ имеются. Причина отсутствия таких супер-ПЭВМ на рынке заключается в том, что, с одной стороны, существующие методы распределенных вычислений не гарантируют реальной производительности при большом числе процессоров для достаточно широкого круга задач, а с другой возникают проблемы использования существующего программного обеспечения.

Для обеспечения высокой надежности вычислительных систем обычно используются различные методы резервирования, что увеличивает необходимый объем аппаратуры в несколько раз. В случае распределенных вычислений высокая надежность может обеспечиваться за счет динамической реконфигурации системы при отказе отдельных элементов при относительно небольшой начальной избыточности. Причем чем больше элементов входит в систему, тем меньшая избыточность требуется для обеспечения необходимой надежности. Однако указанный подход может быть реализован только при полностью децентрализованном управлении, наличии эффективных средств автоматического контроля и диагностики неисправностей, а также средств восстановления ут-

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

раченной в результате ошибок информации. Хотя информация о высоконадежных вычислительных системах, как правило, закрыта в силу использования их в областях, связанных с безопасностью государства (военные системы, энергетика, включая атомную, химические производства и т.д.), есть определенные основания считать, что роль распределенных вычислений в обеспечении надежности таких систем не так велика, как могла бы быть.

Итак, следует признать, что в современной вычислительной технике возможности распределенных вычислений используются достаточно слабо. В то же время Природа демонстрирует нам блестящие образцы высокоэффективных распределенных вычислений, где задачу одновременно решают не тысячи, а миллиарды элементарных вычислителей (нейронов). В результате, хотя скорость работы нейронов в миллиарды раз ниже, чем у современных микропроцессоров, мозг человека способен решать задачи, которые не по силам самым мощным современным супер-ЭВМ. Одновременно и высокая надежность биологических систем обеспечивается без явного резервирования.

Основная причина столь низкой эффективности современных распределенных вычислений заключается в том, что эти вычисления, так же как и обычные последовательные вычисления, как правило, основаны на понятии алгоритма, которое определяется как *последовательность* действий, которую необходимо выполнить для решения задачи. Поскольку распределенные вычисления подразумевают определенную одновременность действий, алгоритм необходимо *распараллеливать*, то есть явно указывать те его части, которые могут выполняться одновременно, что, вообще говоря, противоречит самому определению алгоритма. Каждая часть алгоритма, способная к самостоятельному выполнению в отдельном вычислителе, должна быть оформлена в виде некоторой структуры, обычно называемой процессом. Процесс включает в свой состав как программу, соответствующую алгоритму, так и данные, необходимые для его работы, а также соответствующие результатам работы. Для организации взаимодействия между процессами должны иметься соответствующие каналы связи, а сами процессы должны включать в свой состав механизмы ожидания, обеспечивающие синхронизацию взаимодействия процессов. Таким образом, для организации распределенных вычислений исходный алгоритм должен быть преобразован в сеть, напоминающую сеть потоков данных.

Распределение процессов по отдельным вычислителям и организацию взаимодействия между процессами осуществляет операционная система, однако распараллеливать алгоритм должен программист и от эффективности этого распараллеливания во многом зависит реальная скорость решения задачи. В частности, можно создать большое число процессов, которые будут преимущественно простаивать в ожидании данных от одного или нескольких других процессов. Если объем вычислений, приходящийся на процесс, является небольшим, то может оказаться, что временные затраты на организацию этого процесса, его пересылку в конкретный вычислитель и организацию обмена данными существенно превзойдут собственное время вычислений, в результате чего реальная скорость решения задачи может даже уменьшиться по сравнению с обычным последовательным вычислением. Таким образом, помимо составления программы для задачи, программисту приходится решать весьма непростую оптимизационную задачу распараллеливания, что существенно повышает требования к его квалификации.

Масса проблем возникает и при разработке операционной системы, обеспечивающей распределение процессов по вычислительным устройствам. Даже

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

в простом варианте статического распределения, когда планирование распределения ресурсов осуществляется перед началом решения задачи, для обеспечения относительно полной загрузки оборудования необходимо знать основные параметры процессов, включая оценку времени выполнения и временные характеристики обмена данными с другими процессами. Однако получение таких характеристик на основе исходного алгоритма не всегда представляется возможным. К тому же для многих задач нельзя заранее определить, какие процессы возникнут в ходе вычислений. Динамическое же распределение ресурсов, осуществляемое зачастую в условиях дефицита времени и отсутствия достоверной информации о текущем состоянии ресурсов, требует существенно более сложной реализации.

Таким образом, в рамках существующих подходов распределенные вычисления предъявляют повышенные требования как к программистам, так и к операционным системам по сравнению с обычными последовательными вычислениями. В данной работе рассматривается подход к организации распределенных вычислений, который, наоборот, обеспечивает меньшие требования как к программистам, так и к операционным системам.

2. Динамические автоматные сети

Как известно, для последовательных вычислений существует простая, но исключительно мощная теоретическая модель — машина Тьюринга [1]. Эта машина состоит из конечного автомата и бесконечной ленты, разбитой на последовательно пронумерованные ячейки. Автомат может перемещать ленту в любом направлении, читать содержимое ячеек, соответствующее символам некоторого конечного алфавита, и записывать в них новые символы. Задача считается решенной, когда автомат останавливается. Содержимое ленты после остановки автомата соответствует ответу задачи. Взаимодействие машины Тьюринга с внешним миром, обеспечивающее ввод начальной программы на ленту или вывод ответа задачи, может выполняться тем же автоматом. Очевидно, что большинство современных ЭВМ с одним устройством управления, включая конвейерные или матричные процессоры, легко сводится к машине Тьюринга. Однако адекватно описать распределенные вычисления данная модель неспособна.

Большинство теоретических моделей для параллельных (распределенных) вычислений соответствует направленным графам (операторным сетям), в узлах которых находятся операторы, выполняющие вычисления, а по дугам перемещаются данные. Оператор выполняет вычисления, если для него готовы данные. Подобные модели нельзя назвать конструктивными, поскольку они не рассматривают такие важные для организации вычислений вопросы, как распределение элементов сети по реальным ресурсам; организацию взаимодействия с внешним миром, включая ввод начальной сети и вывод результатов; изменение сетей в ходе вычислений, например при рекурсивных вычислениях; работу со структурными данными и т.д. Соответственно на базе этих моделей нельзя создать реальные вычислительные машины, не привлекая добавочной информации. Предлагаемая ниже теоретическая модель распределенных вычислений лишена указанных недостатков и по степени конструктивности не уступает машине Тьюринга. Хотя прямых прототипов данная модель не имеет, однако ряд основополагающих идей был заимствован из работ фон

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Неймана [2], Улама [3] и Барзиня [4] по теории самоорганизующихся и растущих автоматов.

Решение задачи на ЭВМ можно свести к моделированию изменений структуры объекта, соответствующего данной задаче. При этом из структуры выделяются объекты, которым сопоставляются исходные данные, в то время как отношения между объектами чаще всего определяются неявно (исключение составляют лишь некоторые базы данных) с помощью алгоритмов (программ), задающих преобразование исходных данных в результат. Структура самого алгоритма обычно слабо связана со структурой моделируемого объекта. Искусственное разделение задачи на данные и алгоритмы, с одной стороны, существенно усложняет подготовку задачи к решению, а с другой отделяет моделируемый объект от вычислителя, изменяющего структуру объекта, в результате чего возникает далеко не тривиальная задача отображения объекта на структуру вычислителя.

Более естественным и простым представляется другой подход, заключающийся в том, что каждому элементу моделируемой структуры сопоставляется конечный автомат. Из множества отношений, определяемых над элементами структуры, выделяется конечное число типов первичных отношений (например, отношение принадлежности элемента множеству, отношение предшествования, упорядочивающее элементы), и таким отношениям сопоставляются межавтоматные связи. В результате, исходной структуре сопоставляется автоматная сеть. Если структура является динамической, то в ее состав входят такие элементы (динамические автоматы — ДА), которые способны осуществлять преобразование этой структуры, а именно введение в структуру новых элементов, включая новые отношения, удаление из структуры элементов, изменение структуры элементов. Соответствующая такой структуре динамическая автоматная сеть (ДАС) является распределенным вычислителем, причем решение задачи на таком вычислителе сводится к изменению структуры этого вычислителя. Решение задачи заканчивается, когда в составе ДАС не остается элементов, изменяющих ее структуру, и такая ДАС и является результатом решения задачи. Правда, возможна ситуация, когда преобразования носят циклический характер и результат определяется в виде воздействия ДАС на внешнюю среду.

Определение ДАС осуществляется на трех основных уровнях - первичном, примитивном и базовом. Первые два уровня используются для теоретического, формального представления вычислительной модели, а базовый уровень может использоваться для практического программирования.

На первичном уровне определяются два основных типа первичных автоматов — первичный объект и первичное отношение. Автоматы имеют конечное число полюсов, посредством которых они могут соединяться друг с другом, причем связи между автоматами могут быть иерархическими и линейными, что соответствует отношениям принадлежности и предшествования. Межавтоматные связи реализуются с помощью коммутационной среды, которая будет рассмотрена ниже. Любой автомат, воздействуя на коммутационную среду, может выполнять ряд простейших операций, таких как соединение со свободным первичным автоматом (свободным считается автомат, не связанный ни с каким другим автоматом), за счет чего обеспечивается рост сети, соединение с заданным автоматом, входящим в сеть, удаление связей, соединение с автоматом другой сети. Эти операции достаточны для того, чтобы обеспечить любые структурные преобразования.

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

На следующем уровне ДАС определяются примитивные объекты и отношения, каждому из которых соответствует некоторая ДАС (или, иными словами, макроавтомат), состоящая из первичных автоматов. В любой структуре выделяется первичный автомат (объект или отношение), с которым ассоциируется структура в целом. Поэтому все, что далее говорится о структуре как таковой, относится к соответствующему первичному автомату.

Примитивными отношениями, с одной стороны, являются общезначимые отношения, семантика которых определяется их названием (например, эквивалентность, имя, аргумент, значение), а с другой — ряд отношений, специфичных для рассматриваемой модели (например, состояние, метка или кратность, которые поясняются ниже). К примитивным объектам относится натуральное число и несколько примитивных операций, таких как слияние структур, поиск в структуре подструктур, совпадающих с образцом, и выделение этих подструктур с помощью меток (одно из примитивных специфичных отношений), извлечение из структуры отмеченных подструктур.

На базе объектов и отношений примитивного уровня строятся структуры базового уровня, и при этом формально определяется семантика этих структур. Базовый уровень включает в себя обычный набор числовых и символьных структур, характерных для языков высокого уровня (целые и дробные числа с фиксированной и плавающей запятой в различных системах счисления, комплексные числа, векторы, матрицы, символьные строки и т.д.), стандартный набор арифметических и логических операций, а также элементарных функций, определенных не только над отдельными числами, но и над произвольными структурами, и ряд специфичных операций и отношений, обеспечивающих структурные преобразования и конструирование новых элементов из более простых. В целом базовый уровень ДАС соответствует функциональному языку очень высокого уровня.

В процессе своего развития ДАС может формировать из некоторых своих частей отдельные независимые сети. В то же время рассмотренные выше функции связей не позволяют выполнить обратное действие, то есть объединить несколько различных ДАС в одну сеть. Однако существуют ситуации, когда такое объединение является необходимым, например в случае если разделение некоторой ДАС произошло в из-за ошибки. Учитывая тот факт, что рассматриваемые автоматы являются конечными, а множество автоматов может быть бесконечным, скорее всего решение данной задачи может быть лишь вероятностным. Соединение каналов двух различных автоматов возможно лишь в том случае, если оба канала являются свободными. Если для автомата инициирующего связь, это условие всегда выполнимо, то состояние соответствующего канала автомата, с которым устанавливается связь, заранее неизвестно и может быть определено лишь непосредственно в ходе установки связи. При этом надо учитывать, что несколько автоматов независимо друг от друга могут одновременно пытаться установить связь с одним и тем же каналом, так что даже если к моменту установки связи канал был свободен, связь с ним может установить лишь один из претендентов. Назовем автомат, устанавливающий связь, источником связи, а автомат, с которым устанавливается связь, приемником связи. Легко убедиться в том, что связи между автоматами должны быть двунаправленными, то есть как у источника связи, так соответственно и у приемника для установки связи должен использоваться по меньшей мере один выходной канал и по меньшей мере один входной канал. Действительно, если источник связи использует при установке связи лишь выходной канал, то у него

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

нет возможности получить от приемника информацию о состоянии входного канала приемника, с которым устанавливается связь. Если же источник использует только входной канал, у него нет возможности сообщить приемнику о том, что с ним устанавливается связь.

Использование двунаправленных связей между автоматами дает возможность отказаться от общего для всего множества автоматов времени и считать, что каждый автомат имеет свои собственные часы, не синхронизированные с часами других автоматов. Поэтому далее будем считать, что обмен информацией между двумя любыми автоматами ДАС осуществляется асинхронно.

Любую ДАС можно рассматривать как виртуальную параллельную машину, если каждому автомату ДАС сопоставить некоторый программный элемент. Здесь, так же как и в машине Тьюринга, под программным элементом может пониматься либо некоторая часть программы, либо часть данных, то есть в данном случае программа не делится на две разнородные части, соответствующие собственно программе и данным, а рассматривается как единое целое.

Любая виртуальная машина отражает наиболее существенные с точки зрения программиста черты соответствующей реальной машины, то есть определяет архитектуру этой машины. Поскольку структура виртуальной машины динамически изменяется в процессе ее функционирования, то соответствующая реальная машина имеет динамическую архитектуру.

Для того чтобы приблизить виртуальную машину к реальной, введем некоторые дополнения и уточнения в рассмотренную модель. Рассмотрим некоторое, возможно бесконечное, множество автоматов ввода-вывода. У каждого из этих автоматов часть каналов используется для связи с внешней средой, а другая — для связи с ДАС. Автомат ввода-вывода считается свободным, если он не связан ни с одной ДАС. Свободные автоматы ввода-вывода включаются в область определения функции связей любого автомата ДАС на тех же основаниях, что и другие автоматы. Структура автоматов ввода-вывода в рамках ДАС не определяется. Единственное требование, предъявляемое к этим автоматам, заключается в том, что каналы, с помощью которых эти автоматы связываются с ДАС, должны иметь ту же структуру, что и соответствующие каналы автоматов из основного множества.

Перейдем теперь к рассмотрению взаимодействий между автоматами ДАС в процессе функционирования сети.

Пусть имеется некоторая начальная ДАС, состоящая из автомата $\boldsymbol{a}_0$ и автомата ввода-вывода. В автомат $\boldsymbol{a}_0$ из автомата ввода-вывода поступает линейный текст, который интерпретируется автоматом $\boldsymbol{a}_0$ как древовидная структура. Естественно, что вводимый текст должен содержать средства, позволяющие выполнить подобную интерпретацию.

Каждое слово вводимого текста соответствует программному элементу (ПЭ). Существует два основных вида ПЭ: терминальные, не содержащие ПЭ более низких уровней, и нетерминальные. Между ПЭ существуют как многоуровневые связи, отражающие древовидную структуру программы, так и одноуровневые связи, характеризующие сетевые свойства программы.

Каждому ПЭ сопоставляется свой автомат в ДАС, причем древовидные связи между ПЭ реализуются непосредственно при вводе программы, а сетевые — в ходе выполнения программы. При вводе первого слова текста автомат $\boldsymbol{a}_0$ порождает автомат $\boldsymbol{a}_1$, в который далее это слово передается. Если указан-

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ное слово соответствует нетерминальному ПЭ, то автомат $\boldsymbol{a}_1$ порождает автомат $\boldsymbol{a}_{11}$, а автомат $\boldsymbol{a}_0$ переносит связь от автомата $\boldsymbol{a}_1$ к автомату $\boldsymbol{a}_{11}$, в который далее передается второе слово текста. Если это слово также соответствует нетерминальному ПЭ, то процедура повторяется, т.е. автомат $\boldsymbol{a}_{11}$ порождает автомат $\boldsymbol{a}_{111}$, который соединяется с автоматом $\boldsymbol{a}_0$ и принимает третье слово текста. Если же второе слово соответствовало терминальному ПЭ, то автомат $\boldsymbol{a}_0$ переключает связь на предыдущий уровень — к автомату $\boldsymbol{a}_1$, далее, в зависимости от следующих за этим словом разделителей либо автомат $\boldsymbol{a}_1$ порождает автомат $\boldsymbol{a}_{12}$ и ввод продолжается, либо ввод заканчивается и автомат $\boldsymbol{a}_0$ отключается от автомата $\boldsymbol{a}_1$, порождая тем самым новую независимую ДАС.

После окончания ввода ДАС продолжает изменять свою структуру без какого-либо внешнего управления. Каждый автомат ДАС может присоединять к себе свободные автоматы, обеспечивая тем самым рост ДАС, либо полностью отключаться от других автоматов, становясь свободным и уменьшая число элементов ДАС. Выполнение программы заканчивается, когда ДАС теряет способность к автотрансформации. В этом случае оставшаяся ДАС соответствует ответу задачи. С помощью автоматов ввода-вывода ответ может воздействовать на внешнюю среду и в ходе этого воздействия распасться на свободные автоматы. Нетрудно в поведении ДАС увидеть аналогии с поведением машины Тьюринга.

Можно показать, что рассмотренная выше модель является универсальной, поскольку в ней можно представить любую из существующих универсальных моделей. При этом модель отличается высокой динамичностью, способностью изменять свою программу в ходе вычислений и достаточной конструктивностью, то есть может быть легко реализована.

Главной отличительной чертой динамических автоматных сетей является то, что они рассматриваются как семантические, а не операторные сети, то есть их элементами являются, помимо операторов, данные, представленные в явной форме в виде узлов сети. Кроме того, в качестве элементов ДАС могут использоваться такие объекты, как отношения, ссылки, ресурсы, типы и структуры (подсети). Вычисления в ДАС сводятся к преобразованию всех элементов сети, а не только данных, как в операторных сетях.

Введение в состав ДАС отношений, с одной стороны, обеспечивает более естественное представление задачи в виде сети, а с другой — эти отношения являются очень мощным и гибким средством определения свойств элементов ДАС и задания типов преобразований в сети. В частности, отношения типа «аргумент» определяют управление от данных (это основной способ управления в ДАС). Отношение предшествования, заданное над операторами, позволяет установить последовательность выполнения операторов, определяя тем самым последовательное управление. Отношение принадлежности, заданное над операторами, обеспечивает рекурсивное управление в ДАС. В ходе вычисления отношения могут изменяться и тем самым динамически изменяется и управление.

Ссылки позволяют отмечать в ДАС множества элементов без явного их выделения. Например, ссылками являются кванторы **ВСЕ** или **ЛЮБОЙ,** индекс (номер) элемента в ДАС или диапазон номеров, образец структуры элементов, набор отношений, связанных с элементами. Аппарат ссылок позволяет исключить из ДАС понятие циклов.

Любая ДАС функционирует только в определенной вычислительной среде. В каждый момент времени каждый автомат ДАС размещается в определен-

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ном элементе среды. Как правило, элементы среды не являются одинаковыми и каждый из этих элементов может выполнить лишь одну из функций автомата. Например, такими элементами могут быть различные запоминающие устройства (ОЗУ, магнитные диски, лазерные диски, магнитные ленты и т.д.), выполняющие функции хранения состояния и описания автомата, процессоры различных типов, линии связи. Любой автомат в ходе своего функционирования должен перемещаться от одного элемента вычислительной среды к другому. Каждому типу элемента вычислительной среды соответствует определенный автомат, называемый ресурсом. Задачей любого ресурса является доставка ДАС, с которой связан этот ресурс, к физическому устройству, соответствующему ресурсу. Благодаря ресурсам задачи отображения ДАС на конкретные физические устройства (иными словами, задачи распределения) решаются так же как и обычное вычисление.

В любой ДАС можно выделить автомат, который называется «структурой», определяющий эту сеть как единое целое. Этот автомат связан с остальными элементами ДАС либо непосредственно, либо косвенно отношением принадлежности, то есть остальные элементы ДАС непосредственно или косвенно входят в состав указанной структуры. Основной функцией структуры является перенос связанных с ней отношений на элементы, принадлежащие структуре.

Любой элемент ДАС может характеризоваться состоянием (точнее, любому элементу может соответствовать натуральное число, связанное с этим элементом отношением состояния). Среди состояний можно выделить неопределенное, частично определенное, пустое и безразличное. Неопределенное состояние имеет тот же смысл, что и в любых потоковых моделях, где оператор готов к выполнению, если определены (находятся в любом состоянии, отличном от неопределенного) все элементы, связанные с оператором какими-либо отношениями, а элемент, соответствующий значению (результату) оператора, но не являющийся аргументом этого оператора, находится в неопределенном состоянии или в безразличном. В отличие от большинства потоковых моделей здесь в принципе допускается использование одной и той же структуры и в качестве аргумента, и в качестве результата. Например, возможно следующее выражение, обычно запрещенное в потоковых моделях:

IF A= 0 THEN B:= B+1; ELSE B:= B-1;

В приведенном выражении двоеточие соответствует отношению «имя», то есть B является именем аргумента и именем результата операторов сложения (B+1) и вычитания (B-1). С этими операторами связан отношением условного предшествования результат оператора сравнения (A=0). Если какой-либо аргумент или предшествующий элемент любого оператора находится в пустом состоянии и сам оператор готов к работе, то результат этого оператора также переходит в пустое состояние, кроме ситуации, когда результат является аргументом этого же оператора. В последнем случае состояние значения не изменяется. Ложное значение, находящееся в отношении условного предшествования к любому оператору, эквивалентно пустому состоянию этого значения. Пустое состояние позволяет уменьшить возможности зависания программ.

Смысл безразличного состояния элемента зависит от отношений, связывающих этот элемент с другими, и от операций, для которых этот элемент является аргументом. Поясним это с помощью небольшого примера:

C: = SEARCH (S,B);
B: ~{~}D;

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Первая строчка означает, что C является именем результата операции поиска в структуре S подструктур, совпадающих с образцом B. Структура с именем B состоит из безразличного объекта, находящегося в безразличном отношении к объекту с именем D. Соответственно результат операции поиска будет включать в свой состав подструктуры, содержащие D и все элементы структуры B, находящиеся в каком-либо отношении к D.

Частично определенной называется такая структура, хотя бы один непосредственный элемент которой не определен или определен частично.

Любой оператор в ДАС определяется как сеть, содержащая элементы примитивного уровня и ранее определенные элементы базового уровня. При этом определяется не только семантика оператора, но и допустимые структуры аргументов, которые по отношению к оператору могут быть терминальными и нетерминальными. Если хотя бы один из аргументов оператора является для него нетерминальной структурой, то оператор размножается, порождая множество таких же операторов, аргументами которых являются непосредственные аргументы нетерминальных структур. Эти потомки оператора формируют свои значения как непосредственные элементы структуры, являющейся результатом основного оператора.

Любой оператор, после того как он сформировал результат, разрушается и одновременно разрушает свои аргументы при условии, что они не являются аргументами других операторов и не находятся в состоянии "константа". В последнем случае удаляется только само отношение "аргумент". Существует, однако, возможность многократного применения элементов структуры, определив над ними отношение "кратность". Это необходимо, в частности, при организации циклов.

Важной особенностью предлагаемой модели является возможность динамического изменения отношений (связей) в ходе вычислений. Например, перенос связи от структуры к подструктуре, положение которой вычисляется (в простейшем случае вычисляется индекс элемента массива), либо соединение оператора, выбираемого из заданной структуры по вычисляемому индексу или ассоциативному признаку, с аргументами и значением (реализация оператора CASE), либо оперативное вмешательство в выполнение программы и контроль за ходом ее выполнения. Например, непосредственно в ходе вычисления можно соединить любой элемент программы с оператором вывода на заданное внешнее устройство и оператором ввода, реализуя тем самым диалоговое взаимодействие с программой, что особенно важно при отладке программ.

Явное представление в ДАС данных и отношений в виде автоматов приводит к тому, что модель не нуждается в таком понятии, как память, без чего не обходится ни одна другая вычислительная модель.

Итак, ДАС является столь же конструктивной вычислительной моделью, что и машина Тьюринга, но если последняя представляет последовательные, алгоритмические вычисления, то ДАС обеспечивает распределенные функциональные вычисления. В определенном смысле ДАС является более “мощной”, чем машина Тьюринга, если под мощностью понимать количество вычислений в единицу времени. Мощность машины Тьюринга в указанном смысле определяется числом состояний управляющего автомата и объемом выходного алфавита машины (количество различных символов, записываемых на ленту). Обе эти величины для машины Тьюринга являются конечными, и соответственно конечной является и мощность вычислений у любой конкретной машины Тьюринга. Бесконечной является лишь лента, соответствующая памяти машины и

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

определяющая потенциально бесконечный объем вычислений. В то же время ДАС имеет бесконечно большое число состояний автомата, причем рост числа этих состояний может быть экспоненциальной функцией от времени, так как каждый автомат ДАС может независимо размножаться. Поэтому мощность ДАС может бесконечно увеличиваться со временем в отличие от конечной мощности машины Тьюринга.

Любая машина Тьюринга может быть без труда представлена в виде ДАС, состоящей, например, из линейной цепочки автоматов, соответствующей бесконечной ленте, и автомата, способного перемещаться вдоль этой ленты путем порождения своего потомка влево или вправо с последующим самоуничтожением. Однако никакая машина Тьюринга не в состоянии представить произвольную ДАС. В силу того что доказательство одной из наиболее фундаментальных теорем информатики (сравнимой, пожалуй, лишь с теоремой Геделя) о невозможности решения проблемы остановки машины Тьюринга основано на эквивалентности любых машин Тьюринга, появление машины, более мощной, делает это доказательство некорректным и соответственно вопрос о неразрешимости проблемы остановки машины Тьюринга вновь становится открытым для исследований. Напомним, что указанная проблема имеет глобальное прикладное значение, так как она эквивалентна проблеме возможности решения любой задачи с помощью некоторого алгоритма за конечное время (проблема разрешимости алгоритма).

Сравнивая ДАС с остальными известными вычислительными сетями (операторные сети, сети потоков данных, семантические сети и т.д.), можно заметить, что только ДАС не требуют никакого внешнего управления для своего функционирования и соответственно только ДАС реализуют полностью распределенные вычисления.

Существующие средства вычислительной техники (процессоры, запоминающие устройства) по своей архитектуре ориентированы на традиционную алгоритмическую модель вычислений. Распределенные вычисления могут появляться лишь при объединении этих средств в системы и сети, которые в состоянии решать определенные классы задач и в рамках существующих вычислительных моделей при наличии элементов централизованного управления. В этих условиях достоверная оценка эффективности ДАС может быть получена лишь на основе большого экспериментального материала. Однако если немного заглянуть в будущее, то можно увидеть такую область вычислительной техники, где ДАС будет вне конкуренции. Речь идет о молекулярных или белковых вычислительных машинах, у которых относительно низкая скорость элементной базы компенсируется количеством элементов. Фактически молекулярные процессы на клеточном уровне полностью соответствуют описанным выше механизмам, чего нельзя сказать ни об одной из известных вычислительных моделей.

Тем не менее весь последующий материал посвящен реализациям ДАС в рамках традиционных вычислительных средств.

3. Виртуальная машина с динамической архитектурой

Под виртуальной машиной с динамической архитектурой будем понимать такую конструктивную вычислительную модель, ориентированную на определенную элементную базу (в данном случае микроэлектронную), которая отражает все существенные черты ДАС, но не связана с конкретной реализацией. В

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

самом общем виде МДА представляет собой произвольную коммутационную сеть неограниченных размеров, к которой подключены устройства, реализующие динамические автоматы. Поэтому структурно МДА не отличается от любой вычислительной сети. Отличия появляются на уровне структуры устройств, обеспечивающих вычисления и в большей степени на уровне организации управления вычислениями.

Наиболее естественным способом создания МДА является реализация множества базовых ДА на основе единой интегральной схемы (ИС), которую назовем вычислительным модулем (ВМ). Современная технология позволяет разместить на одном кристалле несколько тысяч ДА. Учитывая высокий уровень повторяемости элементов ИС и возможность нормальной работы даже при наличии определенного числа дефектных ДА, стоимость такой ИС может быть достаточно низкой. Соответственно вполне реальной является МДА, содержащая десятки миллионов ДА и отличающаяся как сверхвысокой вычислительной мощностью, так и сверхвысокой надежностью при полностью децентрализованном управлении. Помимо ВМ в состав машины должны включаться интерфейсные модули (ИМ), обеспечивающие связь с внешним миром, и модули памяти (МП), связанные с запоминающими устройствами большой емкости и обеспечивающие хранение пассивных ДА. Оба этих модуля, так же как и ВМ, содержат множество ДА, но в отличие от ВМ часть ДА данных модулей непосредственно связана с внешними выводами ИС, что позволяет им напрямую взаимодействовать с внешними и запоминающими устройствами. Следует заметить, что полноценная проверка эффективности такой МДА для задач различных классов может быть выполнена лишь на машине достаточно большого объема, что требует больших первоначальных затрат, включающих разработку нескольких типов ИС и изготовление большого их числа. Поэтому целесообразно рассмотреть возможности создания МДА на базе более традиционных вычислительных средств, таких как обычные процессоры.

Рассмотрим ситуацию, когда вычислительный модуль, состоящий из множества ДА, реализован в виде процессора, имеющего память и несколько коммуникационных каналов. В каждом ДА можно выделить три основных компонента: описание, состояние и связи. Описание определяет функционирование автомата и включает в свой состав функции переходов, выходов и функции связей. Состояние и связи не требуют дополнительных пояснений. Если зафиксировать в некоторый момент времени все три компонента автомата и разместить их в определенной области памяти некоторого процессора, то можно говорить о размещении самого ДА в памяти. Без существенного ограничения общности можно считать, что все ДА требуют для своего размещения одинакового объема памяти, поскольку, если какому-либо компоненту ДА недостаточно отведенного для нее места, часть этого компонента может быть представлена в виде самостоятельного ДА, связанного с исходным и используемого только для хранения указанной части компонента. Таким образом, память процессора разбивается на одинаковые участки (ячейки), каждый из которых соответствует одному ДА. Пустая ячейка, не содержащая связей и описания, соответствует свободному ДА. Связи ДА соответствуют либо адресам ячеек памяти, в которых размещаются ДА, связанные с данным автоматом, либо номерам каналов (и подканалов) процессора, либо номерам соответствующих автоматов. Описание ДА является совокупностью программ, выполняемых процессором при реализации функций ДА, а состояние ДА соответствует внутренним переменным этих программ. Следует заметить, что при выполнении функций ДА использу-

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ется и, возможно, изменяется не только состояние самого ДА, но и состояния всех ДА, непосредственно связанных с данным, которые рассматриваются как внешние переменные.

Любой ДА может находиться в пассивном или активном состоянии. Автомат, находящийся в пассивном состоянии не выполняет никаких действий и, следовательно, не нуждается в услугах процессора. Ему необходимо иметь лишь место в памяти. Все свободные автоматы всегда находятся в пассивном состоянии. Активное состояние ДА определяет необходимость выполнения некоторых действий, которые может осуществить только процессор. Автомат переходит из пассивного состояния в активное при любом изменении состояний ДА непосредственно с ним связанных, либо при изменении его собственных связей (например, когда с данным ДА связывается другой автомат или, наоборот, разрушается связь ДА по инициативе другого автомата).

Все активные автоматы становятся в очередь к процессору. Процессор берет очередной ДА из очереди и выполняет функцию, определяемую описанием и состоянием автомата. Если при этом изменяется состояние данного ДА таким образом, что оно может повлиять на другие ДА, процессор по связям находит все автоматы, непосредственно связанные с данным, изменяет их состояние и ставит к себе в очередь. При установке или разрушении какой-либо связи в очередь также ставится автомат, соответствующий этой связи.

Все функции, выполняемые ДА, можно разделить на три основные группы: управляющие, коммутационные и исполнительные.

Управляющие функции соответствуют вычислению некоторых предикатов над состояниями соседних автоматов и определяют переход автомата либо снова в пассивное состояние, либо к выполнению коммутационных или исполнительных функций. Изменение состояния ДА в результате выполнения этих функций не отражается на соседних ДА.

Коммутационные функции обеспечивают передачу информации о состоянии автоматов по связям и изменение связей. Перемещение автомата из одного вычислительного модуля в другой осуществляется путем последовательного выполнения ряда коммутационных функций. Сначала нуждающийся в перемещении автомат через коммутационную сеть устанавливает связь со свободным автоматом заданного типа в каком-либо другом ВМ (порождает новый автомат). Далее в этот автомат передается информация о состоянии, описании и связях текущего ДА, на основании которой автомат может создавать определенную структуру, соответствующую исходной. После завершения передачи текущий ДА разрывает все свои связи и становится свободным (уничтожается). Коммутационные функции можно разбить на три основные группы:

- функции изменения связей и передачи информации по связям внутри ВМ (внутренняя коммутация);
- функции изменения связей и передачи информации между ВМ (внешняя коммутация);
- функции взаимодействия с внешней средой (ввод-вывод).

Исполнительные функции обеспечивают изменение собственного состояния ДА и формирование выходных сигналов, отражающих изменение состояния. При реализации данных функций не участвуют в явном виде состояния соседних автоматов, поскольку к началу выполнения этих функций указанные состояния должны быть отображены с помощью коммутационных функций в собственное состояние ДА.

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

Каждая из указанных выше функций автомата требует для своей реализации специфичных технических средств. Так, для исполнительных функций эти средства должны предоставлять широкий набор арифметических и логических операций. Управляющие функции сводятся к вычислению предикатов и потому требуют эффективных средств работы с полями логических данных. Коммутационные функции нуждаются в развитых средствах адресации и специальных командах, обеспечивающих взаимодействие с КАС и внешним миром. Объединение этих средств в одном процессоре приводит к низкому коэффициенту загрузки его аппаратуры, поскольку в каждый момент времени реализуется лишь одна из функций. Поэтому целесообразно представить вычислительный модуль в виде многопроцессорной системы с функционально-ориентированными процессорами.

Каждый процессор имеет локальную память, в которой активные автоматы, нуждающиеся в функциях данного процессора, выстраиваются в очередь. Кроме того, имеется главная общая память ВМ, где находятся пассивные и свободные автоматы. Эта память доступна управляющему процессору, выполняющему также и внутренние коммутационные функции. Каждый процессор берет из очереди автомат, выполняет соответствующую функцию и помещает его в очередь к другому процессору или снова в свою очередь. Автомат, переходящий в пассивное состояние, помещается в главную память ВМ (этим занимается управляющий процессор). Главная память не обязательно соответствует только оперативной памяти с произвольным доступом. В ее состав может входить память на гибких или жестких дисках, доступ к которой осуществляется через соответствующие контроллеры, управляемые УП.

Как уже отмечалось выше, вычислительные модули могут объединяться в структуры произвольного вида. Элементы любой вычислительной сети могут рассматриваться как вычислительные модули, а сама сеть — как МДА при условии, что в этих модулях реализованы рассмотренные выше механизмы управления и коммутации. Однако при специальном проектировании МДА целесообразно использовать иерархические (кластерные) структуры. Такие структуры обладают неограниченными способностями к росту, в то время как длина связей растет как логарифм от общего числа элементов МДА.

Предположим, что коммутационный процессор вычислительного модуля имеет 4 внешних канала связи. Тогда четыре ВМ могут объединиться в кластер первого уровня МДА1 таким образом, что три канала связи каждого ВМ используются внутри кластера, а остальные каналы (по одному у каждого ВМ) являются внешними связями кластера. Таким образом, с точки зрения связей ВМ и кластер эквивалентны. Аналогично четыре МДА1 объединяются в кластер следующего уровня МДА2, у которого тоже имеется четыре канала внешних связей, и т.д.

В рассмотренном варианте МДА мощность связей (число физических линий, образующих связь) на любом уровне МДА остается неизменной. Однако если число каналов связи каждого ВМ превосходит число элементов кластера, мощность связей будет расти при переходе на более высокие уровни кластеров. Например, если ВМ имеет не 4, а 6 каналов связи, то при той же структуре МДА1 имеет 12 внешних каналов вместо 4. Если половина этих каналов используется для внутренних связей в МДА2, то число внешних связей МДА2 равно 24. МДА уровня n имеет в своем составе 4^n вычислительных модулей и $6*2^n$ внешних каналов связи, причем мощность любой внутренней связи между

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

элементами этой МДА равна 2^{n-1}. Таким образом, при увеличении уровня кластеров растут и мощности межкластерных связей.

Структура МДА не обязательно должна иметь строго регулярный характер. Так, кластеры различных уровней могут иметь различное число элементов, причем в состав кластера уровня ***n*** могут непосредственно входить не только кластеры уровня ***n-1***, но и кластеры более низких уровней. Причем любой кластер имеет уровень, на единицу превышающий максимальный из уровней его непосредственных элементов.

В состав МДА могут входить вычислительные модули различных типов, отличающиеся прежде всего структурой исполнительных процессоров и процессоров ввода-вывода, а также типом и объемом общей памяти. Каждому процессору или устройству памяти сопоставляется специальный ДА, называемый ресурсом. Доступ к различным ресурсам осуществляется с помощью соответствующих индикаторов, причем однотипным ресурсам соответствуют групповые индикаторы.

Любой автомат еще на стадии написания программы может быть связан с одним или несколькими ресурсами заданного типа. Эти связи могут определяться явно или наследоваться от автоматов, соответствующих структурам, содержащим данный автомат, или задаваться по умолчанию. Кроме того, связи с ресурсами могут определяться непосредственно в ходе работы автомата. Ресурсы могут относиться как к автомату в целом (например, память определенного объема и способа доступа), так и к отдельным функциям автомата (процессоры различного типа). Как сами ресурсы, так и отношения между ресурсами и различными автоматами структурно не отличаются от любых других ДАС. Поэтому возможно неограниченное разнообразие как самих ресурсов, так и их взаимодействия с автоматами других типов. Например, автомату может соответствовать множество различных ресурсов для одной и той же функции, и выбор конкретного ресурса определяется динамически в зависимости от состояний как самого автомата, так и некоторых других автоматов.

Основной функцией ресурсов является доставка ДА, с которыми эти ресурсы связаны, к тем физическим устройствам, которым ресурсы соответствуют. Поэтому при переходе ДА в активное состояние или при первичном появлении ДА в ВМ происходит активация соответствующего ресурса, который определяет наличие в данном ВМ соответствующих ресурсу физических устройств, а при отсутствии последних обеспечивает транзит ДА в модуль, содержащий эти средства. Задачей ресурса является также выполнение определенных действий при отсутствии в МДА в целом указанных устройств или при монопольной занятости этих устройств.

Одним из основных типов ресурсов являются очереди, обеспечивающие взаимодействие ДА с процессорами как на уровне отдельных ВМ, так и МДА в целом, поскольку именно очереди решают проблему выбора между несколькими однотипными процессорами, соответствующими одному ресурсу.

Очередь в МДА представляет собой ДАС, определяющую порядок взаимодействия множества ДА с некоторым ресурсом или множеством однотипных ресурсов. Каждый автомат по отношению к определенному ресурсу обладает некоторым приоритетом. Приоритет может присваиваться в ходе написания программы или динамически в ходе ее выполнения. Автомат, обладающий некоторым высоким приоритетом, может временно присваивать его другим ДА, с которыми он связан. В частности, автоматы, переходящие в активное состояние в результате воздействия внешних сигналов (через автомат

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ввода-вывода) или в результате окончания некоторого временного интервала (с помощью автомата, реализующего таймерные функции), как правило, получают более высокий приоритет по отношению к другим автоматам. Одноприоритетные автоматы выполняются в порядке их поступления в очередь (первый пришел — первый ушел), однако при наличии в очереди автоматов с различными приоритетами сначала обслуживаются ДА с более высоким приоритетом независимо от времени их поступления в очередь.

Для каждого приоритета у очереди существует лимит числа элементов. При достижении этого лимита хотя бы по одному приоритету может осуществляться реструктуризация очереди. При наличии в МДА хотя бы одного незанятого ресурса того же типа, которому соответствует очередь, исходная очередь порождает двух потомков, один из которых соответствует старой очереди к ресурсу в данном ВМ, а второй — очереди к ресурсу в другом ВМ, с которым устанавливается связь и в который посылается копия исходной очереди. При наличии нескольких очередей к ресурсам различных ВМ автомат ставится в ту очередь, которая имеет меньше элементов, однако более приоритетные автоматы ставятся в очередь к ресурсу в текущем ВМ. Таким образом, происходит динамическое распределение задачи по модулям ВМ.

Обычно в устройствах локальной памяти процессоров размещаются копии автоматов, которые сами находятся в главной памяти модуля. Эти же копии посылаются в другие ВМ при размножении очередей к исполнительному процессору или процессору ввода-вывода. После завершения выполнения такой копии автомата она возвращается в исходный ВМ, где находится основной автомат. Только после этого соответствующий элемент удаляется из исходной очереди. Каждая очередь осуществляет контроль времени пребывания в ней элементов. Если в какой-то очереди в течение определенного времени не удаляются элементы, то все стоящие в этой очереди автоматы передаются в другие очереди к таким же ресурсам (при этом изменяются номера копий), а к данному ресурсу посылается контрольно-диагностический автомат, который обращается к ресурсу помимо любых очередей, определяет исправность этого ресурса и пытается привести его в рабочее состояние. Таким образом, гибель копии автомата в пути или в другом ВМ не приводит к потере информации. Если же автомат просто задержался в дальней очереди и все-таки появляется в исходном ВМ, то он игнорируется, поскольку рабочей уже является другая копия.

Если все очереди к ресурсу заданного типа оказываются полными, процессор, ставящий автоматы в эти очереди, перестает это делать и устанавливает данные автоматы в конец собственной очереди, чтобы вернуться к ним через определенное время. Кроме того, приостанавливаются процессы размножения автоматов (автомат, нуждающийся в размножении также ставится в конец очереди). Таким образом, при переполнении очередей не происходит остановок процессоров или потери автоматов. Следует заметить, что потеря копии автомата в локальной памяти процессора не является фатальной, поскольку сохраняется основной автомат в общей памяти ВМ.

При переполнении очередей к управляющему или коммутационному процессорам, копии автоматов из этих очередей, посылаемые в другие ВМ, размещаются там в общей памяти и становятся основными автоматами, а в исходном ВМ эти автоматы уничтожаются после подтверждения об укоренении копии на новом месте. Таким образом, осуществляется миграция ДАС по МДА.

Пустая очередь к процессору в другом ВМ уничтожается. Пустая очередь к процессору в собственном ВМ приводит к блокировке постановки в очереди к

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

таким же процессорам из других ВМ, с последующих их уничтожением после выполнения всех автоматов из этих очередей.

Помимо копий автоматы могут создавать дублеров. Отличие дублера от копии заключается в том, что дублер размешается в главной памяти другого ВМ по отношению к исходному автомату и независимо от изменения состояний соседних автоматов остается в пассивном состоянии, получая, однако, от основного автомата полную информацию об изменении состояний последнего. Основной автомат независимо от своей активности должен с определенной периодичностью связываться с дублером. При отсутствии информации от основного автомата в течение определенного времени дублер сам пытается связаться с основным ДА и в случае неудачи такой попытки становится основным ДА и создает себе дублера в другом ВМ. Для задач, требующих особенно высокой надежности, возможно одновременное использование нескольких дублеров и нескольких копий основного ДА, связанных общей связью. Каждая копия реализуется в отдельном ВМ, и выполнение некоторой функции ДА завершается после отработки всех копий и совпадения результатов работы (состояний автомата) для большинства копий. Сравнение этих результатов осуществляется основным ДА и каждым из его дублеров при отображении состояний копий после выполнения функции в состоянии исходного ДА и дублеров. Если половина или большее число копий получили различные результаты, то основной ДА вновь порождает копии и посылает их на выполнение к соответствующим процессорам. В случае гибели основного ДА его место занимает дублер с меньшим номером (при создании дублеров основной ДА нумерует их) как старший по возрасту и порождает еще одного дублера. При нормальном самоуничтожении ДА он сначала санкционирует самоуничтожение своих дублеров, убеждается, что связь со всеми дублерами разрушилась (каждый дублер при уничтожении разрушает все свои связи), определяет отсутствие своих активных копий в каких-либо очередях и лишь после этого разрушает свои связи и становится свободным автоматом. Если хотя бы одно из указанных условий не выполнено, ДА связывается со специальным автоматом времени, который имеется в любом ВМ и который обеспечивает активацию связанных с ним автоматов через интервалы времени, определяемые этими автоматами. По истечении определенного интервала времени ДА активируется и разрушается уже безусловно.

Мультипроцессорная структура вычислительных модулей МДА с функционально-ориентированными процессорами обеспечивает не только эффективное использование аппаратуры и высокую реальную производительность, но и высокий уровень информационной защиты от несанкционированного воздействия одних программ на другие. Прежде всего заметим, что из всех процессоров вычислительного модуля только один (управляющий) работает с главной памятью и с оригиналами автоматов. Соответственно только этот процессор может изменять структуру автоматов, изменять связи (возможно, при участии коммутационного процессора), уничтожать или порождать автоматы. Система команд УП, реализованная на аппаратном уровне и не доступная для внешних изменений, не позволяет обращаться к произвольным ячейкам физической памяти независимо от ее типа и допускает только обращение к автоматным структурам. При этом в автомате допускается изменение только состояния и связей, но не описания (кроме момента рождения нового автомата, когда задается его структура и описание). Таким образом, функции автомата при исправном УП не могут подвергнуться изменению. Само измене-

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ние любого ДА в общей памяти осуществляется путем отображения в него состояния и связей копии, после завершения выполнения ее функции в УП. Каждая копия ДА содержит указатель на положение оригинала в главной памяти и индекс оригинала. При несовпадении индекса копии и оригинала копия объявляется ошибочной и ее воздействие на ДА блокируется. Таким образом, ДА может изменяться только под воздействием собственной копии. Если система команд любого процессора ВМ не допускает возможностей изменения отдельных полей в копии (в частности, указателя на ДА) и не позволяет непосредственно обратиться к иным копиям, размещенным в локальной памяти, кроме текущей, то при исправных процессорах исключаются несанкционированные воздействия одних программ на другие.

Однако такой подход ограничивает применение МДА только рамками вновь создаваемых процессоров с оригинальными системами команд, что приводит к практической несовместимости с ранее созданным программным обеспечением. Совместимость можно обеспечить, если использовать в качестве исполнительного процессора (ИП) один из стандартных микропроцессоров, дополненный в случае необходимости специальными аппаратными средствами защиты памяти. В этом случае любая пользовательская программа (последовательного типа) представляется в виде одного ДА. При загрузке этой программы в локальную память ИП, соответствующую оперативной памяти процессора, в ней выделяется два линейных участка, один из которых используется для описания ДА или собственно программы, а второй — для области данных. Схема защиты памяти допускает обращение только к указанным участкам в ходе выполнения программы, причем в первый из участков не разрешается запись. При возникновении в программе обращения к внешним устройствам ИП создает автомат, передаваемый в УП и несущий в себе либо запрос на внешние данные, или недостающие части программы, или информацию, передаваемую во внешние устройства (в общую память модуля, в другие ВМ или во внешнюю среду). При этом ДА, соответствующий выполняемой программе, ставится в конец очереди ИП. В конец очереди ДА ставится и при истечении кванта времени, выделяемого для выполнения любому автомату в ИП.

Итак, несанкционированное взаимодействие программ в МДА практически исключено. Единственным санкционированным взаимодействием является установка связей между ДА. Этот процесс является двусторонним. Сначала автомат ***a***, желающий установить связь с автоматом ***b***, должен каким-то образом сообщить о своем желании автомату ***b*** и передать ему свой индекс и, возможно, пароль для установки связи. Затем, если автомат ***b*** сочтет возможным, он сам устанавливает данную связь (большинство автоматов ДАС допускает установки связей лишь в строго определенных пределах конкретной ДАС). Однако и при установленной связи автомат ***a*** не может непосредственно изменять состояние автомата ***b***, не говоря уже о описании или связях этого автомата, а может лишь косвенно влиять на автомат ***b***, как и любой другой сосед, или использовать состояние автомата ***b*** для изменения своего собственного состояния.

Все выше перечисленное определяет невозможность в МДА умышленной или случайной порчи одних программ другими при исправной аппаратуре и потому обеспечивается высокий уровень иммунной защиты от традиционных программных вирусов. Однако в результате ошибок программистов или неисправностей аппаратуры в МДА могут появляться дефектные ДАС, которые занимают ресурсы и тем самым ограничивают возможности развития нормальных

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

ДАС. Особенно опасными в этом смысле являются “раковые опухоли”, когда автоматы начинают неограниченно размножаться, занимая ресурсы общей памяти ВМ, или гиперактивные ДАС, у которых большая часть ДА находится в активном состоянии, занимая тем самым значительную часть процессорных ресурсов. Для выявления и уничтожения дефектных автоматов каждый ВМ имеет специальные аппаратно реализованные проверяющие ДА (ПДА), которые единственные могут непосредственно обращаться к любым автоматам или их копиям в устройствах главной или локальной памяти и уничтожать эти автоматы. Единого критерия, позволяющего отличить нормальный ДА от дефектного, не существует, как и в биологических или социальных системах, но тем не менее значительную часть дефектных автоматов, мешающих работе нормальных ДА, можно выявить или по меньшей мере ограничить их отрицательное влияние.

4. Ретроспектива и реализация

В 1968 году, будучи доцентом ЛИАП, я получил заказ от одного из предприятий Министерства общего машиностроения (так в то время называлось министерство ракетной техники) на исследование архитектуры вычислительной машины, предназначенной для управления марсоходом. Исходя из условий работы на далекой планете, эта машина должна была сохранять работоспособность в необслуживаемом режиме в течение длительного времени, отличаться малыми размерами и энергопотреблением при относительно большой вычислительной мощности, так как прямое управление с Земли, характерное для луноходов, здесь было невозможно из-за больших расстояний. В рамках известных в то время архитектур решить эту задачу было нельзя даже теоретически, прежде всего потому, что никакое резервирование не защищает от множественных отказов элементов, а последовательный характер вычислений ограничивает вычислительную мощность.

Решение проблемы было подсказано Природой. Вместо одного большого вычислителя надо создать сеть из большого числа примитивных вычислительных устройств – элементарных процессоров (ЭП). Каждый ЭП связан лишь с ближайшими соседями и может настраиваться в зависимости от информации, полученной от соседей на выполнение тех или иных вычислительных, логических, управляющих или коммуникационных функций. Каждый ЭП контролирует состояние ближайших соседей и в случае обнаружения неисправности соответственно изменяет свое поведение, чтобы сеть в целом продолжала функционировать нормально. Для того чтобы максимально упростить процессоры (степень интеграции микросхем в то время была относительно невысока), желательно провести их специализацию по группам функций – вычислительные, управляющие и коммуникационные. Поскольку число ЭП могло быть сколь угодно большим, желательно было иметь общий метод описания подобных структур, состоящих из разнотипных элементов с различными связями между элементами. В качестве такого метода было предложено использовать рекурсивные описания; сам вычислитель получил название «рекурсивная машина» (РМ). В самой общей форме *рекурсивная машина — это множество рекурсивных машин и/или элементарных процессоров, соединенных друг с другом определенным образом.*

Следует заметить, что в то время были весьма популярны (на уровне публикаций) однородные вычислительные структуры. Множество простых вычис-

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

лительных элементов (ВЭ) объединялось в двухмерную или трехмерную структуру с целью повышения производительности. При этом либо все ВЭ одновременно выполняли одну и ту же операцию, но с разными данными, поступающими из центрального устройства управления (матричные структуры), либо каждый ВЭ настраивался центральным устройством на выполнение некоторой функции и находился в этом состоянии длительное время. В обоих случаях предусматривалось наличие центрального устройства управления, которое было узким местом, как в плане обеспечения высокой надежности, так и высокой производительности за исключением узкого класса задач. В рекурсивной машине таких узких мест не было в принципе.

Сетевая структура РМ диктовала и сетевую структуру выполняемых программ. К тому времени появились первые публикации по операторным сетям, управляемым данными *(data driven nets)* [5–7]. Такие сети соответствуют направленным графам, в вершинах которых находятся операторы, обеспечивающие преобразования входных данных в выходные, а дуги используются для хранения и передачи данных. Оператор начинает преобразование, если все его входные данные готовы. Однако непосредственная реализация таких сетей в качестве программ для РМ не представлялась возможной из-за отсутствия механизмов отображения операторной сети на сеть ЭП. Поэтому была предложена рекурсивная модификация операторных сетей за счет введения структурных операторов. *Структурный оператор соответствует направленному графу, вершинами которого являются структурные операторы и/или примитивные (терминальные) операторы*. Выполнение структурного оператора в управляющем процессоре заключается в передаче элементов оператора в процессоры РМ для последующего выполнения, если все операнды этого структурного оператора готовы. Итак, в рекурсивной машине не только структура аппаратных средств, но и структура программ определяется рекурсивно.

В ходе выполнения указанного НИР была детально проработана структура РМ для марсохода и основных процессоров, а также элементов программного обеспечения и сделаны оценки надежности, размеров и энергопотребления РМ. При этом было получено более 10 авторских свидетельств на изобретения, в частности такие как «Рекурсивно-однородная структура» [8], «Регулярное вычислительное устройство» [9], «Вычислительный модуль с перестраиваемой структурой» [10]. Правда, этот проект не был реализован, так как была закрыта программа марсианских полетов.

В 1974 году на конгрессе ИФИП в Стокгольме концепция рекурсивных вычислительных машин была предложена как антитеза машин фон-неймановского типа [11]. В современной терминологии эти машины относились к классу динамических потоковых систем с неограниченным (рекурсивным) структурированием как операторов, так и данных и с произвольным числом элементов, соединенных в рекурсивно определяемые структуры. Этот доклад произвел большое впечатление на участников конгресса. Об этом говорит хотя бы такой факт, что в 1999 году в Нью-Йорке состоялся симпозиум, посвященный 25-летию со дня доклада. В ряде стран начались работы в области рекурсивных ЭВМ [12–15], определив создание соответствующего направления. Ведущая компьютерная фирма США «Control Data Corporation» планировала совместные работы по созданию РМ [16], и лишь холодная война между США и СССР помешала реализации этого проекта. В середине 1979 г. в ЛИАПе был создан небольшой макетный образец рекурсивной ЭВМ производительностью в 12 раз выше, чем у ЭВМ БЭСМ-6, испытания которой в присутствии государст-

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

венной комиссии под руководством академика А.А.Дородницина подтвердили основные положения концепции. После этих испытаний в конце 1979 г. выходит Постановление СМ СССР и ЦК КПСС о развитии работ в области рекурсивных машин. В середине 80-х годов в Киеве под руководством академика В.М.Глушкова (одного из соавторов этого доклада) была создана полномасштабная рекурсивная ЭВМ ЕС-1701 с производительностью порядка 500 млн. операций в секунду.

Результатом этих испытаний явилось решение о создании в ЛНИВЦ АН СССР лаборатории вычислительных структур, которое было принято в конце 1979 г. по инициативе академика–секретаря Отделения механики и процессов управления академика Б.Н.Петрова и первого директора института В.М.Пономарева. Основной задачей лаборатории являлись и являются исследования в области архитектуры вычислительных машин, качественно отличающихся от существующих. Лаборатория создавалась под конкретного руководителя – доцента Ленинградского института авиационного приборостроения к.т.н. В.А.Торгашева с целью развития уже полученных к тому времени результатов.

Анализ недостатков рекурсивных ЭВМ, выявленных в ходе экспериментальных исследований, привел в начале 80-х годов к разработке конструктивной теоретической модели распределенных вычислений — динамическим автоматным сетям [17, 18], на основе которой в середине восьмидесятых годов в Москве на базе НИЦЭВТ под руководством В.А.Торгашева были созданы мультипроцессоры с динамической архитектурой ЕС-2704. Эти мультипроцессоры успешно эксплуатировались в качестве вычислительных систем реального времени и не только показали высокие технические характеристики (производительность — 100–300 млн операций в секунду в зависимости от классов решаемых задач; отношение реальной производительности к максимальной — 0.85–0.92; сохранение работоспособности при множественных отказах), несмотря на устаревшую элементную базу, но и полностью подтвердили основные теоретические концепции машин с динамической архитектурой.

Практическое прекращение финансирования проектов в области супер-ЭВМ и распределенных вычислений, которое произошло в России после 1991 года, не позволило продолжать работы в области крупномасштабных распределенных систем. Однако было создано несколько систем с динамической архитектурой на базе высокопроизводительных микропроцессоров [19], одна из которых, выполненная по заказу американской фирмы Nick&C на 10 микропроцессорах TMS320C40, для решения задач автоматизации проектирования авиадвигателей с использованием трехмерной графики демонстрировалась на международном авиасалоне (г. Жуковский, Россия) в 1995 г. Продолжалась также работа по языковым средствам [20]. Но основные исследования были сосредоточены в области архитектуры вычислительных модулей — процессоров с динамической архитектурой, способных подстраивать систему команд под структуру решаемой задачи. Оказалось, что на базе существующих технологий, в частности схем гибкой логики в сочетании с высокопроизводительными микропроцессорами, можно создавать достаточно универсальные модули, которые не только могут использоваться в качестве элемента распределенной системы, но и эффективно использоваться в качестве акселератора для персональной ЭВМ, повышая на 1–2 порядка эффективную производительность последней при работе, в том числе и в системах реального времени, таких как например радиолокационные или телекоммуникационные системы. Это подтверждается

Труды СПИИРАН. 2007. Вып. 4. ISSN 2078-9181 (печ.), ISSN 2078-9599 (онлайн)
SPIIRAS Proceedings. 2007. Issue 4. ISSN 2078-9181 (print), ISSN 2078-9599 (online)
www.proceedings.spiiras.nw.ru

практикой использования процессоров с динамической архитектурой DAP-311 на нефтяных платформах в Норвежском море в системах радиолокационного наблюдения, обеспечивающих автоматическое обнаружение и сопровождение судов в акватории платформ, а также в ряде портов России в системах управления движением судов [21].

Все современные суперкомпьютеры используют процессоры с традиционной (фон-неймановской) архитектурой, где в каждый момент времени работает лишь очень малая часть аппаратуры. Если взять самые лучшие образцы современных суперкомпьютеров, таких как например «Blue Gene» фирмы IBM, то мы имеем следующие характеристики в расчете на единицу пиковой производительности 1 Тфлопс (триллион операций с плавающей запятой в секунду). Стоимость составляет не менее 200 тысяч долларов США, размеры – не менее 300 дм^3, энергопотребление – не менее 6 кВт. Для реальных задач эти цифры увеличиваются на 1–2 порядка.

При использовании процессоров с динамической архитектурой типа DACM120, основанных на типовых микросхемах фирмы «Altera» и разработанных в СПИИРАН совместно с фирмой «Динамические электронные системы» (Санкт-Петербург, Россия), можно получить существенно лучшие характеристики суперкомпьютера. Стоимость будет составлять не более 5 тысяч долларов США, размеры не более 2 дм^3, энергопотребление – не более 50 Вт. То есть все характеристики лучше в 40–150 раз. На реальных задачах выигрыш может увеличиться еще на порядок. При этом можно иметь широкий спектр суперкомпьютеров, начиная от акселераторов для ПЭВМ с производительностью от 100 Гфлопс до 3.2 Тфлопс и кончая суперкомпьютерами с производительностью более 10 Пфлопс (10 000 Тфлопс) при достаточно небольших размерах.

Небольшие модули с динамической архитектурой в виде платы для обычной ПЭВМ обеспечивают эффективную защиту практически от любых сетевых атак, надежное хранение конфиденциальных данных в распределенных структурах, возможности любому пользователю выполнять распределенные вычисления в сети Интернет, не затрагивая основных ресурсов ПЭВМ, при полной гарантии безопасности.

Литература

1. *Turing A. M.* On computable numbers with an application to the Enstcheiaungsproblem // Proc. London. Math. Soc. 1937. Vol. 43.
2. *J. von Neuman* Theory of self-reproducing automata. University of Illinois Press, Urbana and London, 1966.
3. *Ulam S. M*. Random processes and transformation // Proc. of the International Congress of Mathematicians. 1950, Providence, 1952, vol. 11, p. 264–275.
4. *Барзинь Ю. М.* Проблема универсальности растущих автоматов // Доклады АН СССР 1964 Т. 57, № 3, С. 542–545.
5. *Petri C. A.* Fundamentals of a theory asynchronous information flows // IFIP Conf. Proc. North-Holland Publ. Co. 1962. P. 386–391.
6. *Karp R. M. and Miller R. E.* Properties of a model for parallel computations: Determinacy, Termination, and Queuing // SIAM J. Applied Math. 1966, vol.14, No. 6. P. 1390–1411.
7. *Dennis J. B.* Programming generality, parallelism and computer architecture // IFIP Conf. Proc. North-Holland Publ. Co. 1968. P. 484–492.
8. *Торгашев В. А., Андрианов В. И., Бердников Л. И.* Рекурсивно-однородная структура. А.с. № 1445061 с приоритетом от 19 мая 1970 г.
9. *Торгашев В. А., Андрианов В. И., Бердников Л. И.* Регулярное вычислительное устройство. А.с. № 1499037 с приоритетом от 16 ноября 1970 г.
10. *Торгашев В. А., Андрианов В. И., Кисельников В. М., Смирнов В. Б.* Вычислительный модуль с перестраиваемой структурой. А.с. № 1844632 с приоритетом от 4 ноября 1972 г.

11. *Glushkov V. M., Ignatyev M. B., Myasnicov V. A. and Torgashev V. A.* Recursive machines and computing technology // IFIP Conf. Proc. Amsterdam: North-Holland Publ. Co. 1974. P. 65–70.
12. *Cruzela I.* EMCS — Design of unorthodox multiprocessing computer system // Proc. 2-nd EUROMICRO Symp. Venice. 1976. P. 153–156.
13. *Davis A. L.* The architecture and system methodology of DDM1: A recursively structured data driven machine // Proc. 5-th Annual Symposium Computer Architecture. 1978. P. 210–215.
14. *Wilner W. T.* Recursive machines // Internal Report Xerox Corp. Palo Alto Research Center, 1980.
15. *Treleaven P. C., Hopkins R. P.* A recursive computer architecture for VLSI // Proc. 9-th International Symposium Computer Architecture. 1982.
16. CDC may explore new soviet technology // Computer Digest. 1977. Vol. 12, No. 11. P. 7.
17. *Торгашев В. А.* Управление вычислительными процессами и машины с динамической архитектурой. Вычислительные системы и методы исследований и управления автоматизации. М.: Наука, 1982. С. 172–187.
18. *Плюснин В. У., Пономарев В. М., Торгашев В. А.* Распределенные вычисления и машины с динамической архитектурой. Вычислительные системы и методы исследований и управления автоматизации. М.: Наука. 1982. С. 188–205.
19. *Torgashev V. A. and Plyusnin V. U.* Dynamic architecture computers // Proc. of the Intern. Conf. Parallel Computing Technologies. ReSCo. Moscow, 1993. P. 25–29.
20. *Torgashev V. A. and Tsaryov I. V.* A parallel programming language for dynamic architecture computers // Proc. of The Intern. Conf. Parallel Computing Technologies. ReSCo. Moscow. 1993. P. 31–34.
21. *Afanasyev V. V., Sapegin V. B. and Torgashev V. A.* Dynamic architecture processors in vessel traffic systems // Proc. of the Intern. Conf. on Informatics and Control, ICI&C.97. St.-Petersburg, 1997.