ПРОБЛЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ
(Вычислительные системы)
1983 год Выпуск 100

УДК 681.142.2

О ПУТЯХ РАЗВИТИЯ МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ СРЕДСТВ ВЫЧИСЛЕНИЙ

Ю.Г.Косарев

I. За сорок лет своей истории ЭВМ из экзотического устройства превратилась в одно из основных орудий научно-технического прогресса, которое активно преобразует буквально все сферы человеческой деятельности.

Несмотря на бурный рост индустрии средств вычислений, она все равно не поспевает за стремительно увеличивающимися потребностями практики. Если в предшествующий период развитие вычислительной техники сдерживалось в основном отставанием технологической базы, то сейчас, когда достигнут огромный прогресс в изготовлении дешевых, надежных и быстродействующих микросхем с большой степенью интеграции элементов, все отчетливее видно, что узким местом становится разработка математического обеспечения. И, как это часто бывает, дело здесь не только и не столько в недостатке усилий, сколько в самой научно-технической политике. До сих пор по инерции основные усилия разработчиков следуют в русле исторически сложившейся парадигмы, в основе которой лежит следующее.

I.I. Ориентация на математически подготовленного пользователя: программиста, владеющего машинно-ориентированными языками и способного успешно адаптировать алгоритмы и программы к особенностям технических средств, и пользователя, который, не являясь по профессии программистом, сумел овладеть проблемно-ориентированными языками программирования. Учет особенностей технических средств в этом случае возлагается на соответствующий транслятор.

I.2. Ориентация на однопроцессорные ЭВМ со сравнительно небольшими объемами оперативных памятей.

I.3. Жесткая привязка создаваемых программных средств к определенному (фиксируемому при трансляции или в процессе реализации) объему оперативной памяти, структуре и конфигурации технических средств, количеству (обычно одному) процессоров и т.п.

I.4. Ограничение логической модели, лежащей в основе конструирования языков программирования, моделью последовательной машины фон Неймана. Обычно это ограничение явно не указывается (а может быть, и не всегда осознается) авторами этих языков [I].

I.5. Отсутствие жестких запретов на создание в рамках одной и той же логической модели различного рода вариаций проблемно-ориентированных языков программирования, отличающихся друг от друга непринципиальными деталями (обычно касающихся формы представления данных).

I.6. Отсутствие единого обязательного стандартна на машинно-ориентированные языки программирования, языки общения пользователя с операционными системами, банками данных и т.п.

2. Можно видеть, что существующая парадигма пришла в противоречие с требованиями практики буквально по всем указанным пунктам.

2.I. Повсеместное распространение ЭВМ требует создания простых и доступных для массового неквалифицированного пользователя непроцедурных средств общения с ЭВМ, с помощью которых он мог бы указать, ЧТО он хочет вычислить (и при этом был бы избавлен от необходимости описывать на каком-либо формальном языке самую процедуру вычислений). Расчет на возможность даже в неотдаленном будущем обучить массового пользователя программированию на каком-либо процедурном языке (даже на простейших проблемно-ориентированных языках) равносилен потере чувства реальности.

2.2. Попытки построения программ для многопроцессорных систем в рамках традиционной парадигмы сводились к созданию универсальных средств, которые бы позволяли автоматически распараллеливать программы либо в момент их трансляции, либо в момент их выполнения. К исходным программам при этом не предъявлялись какие-либо дополнительные требования по сравнению с исходными программами для однопроцессорных ЭВМ. Иначе говоря, делались попытки решить проблему распараллеливания на той же основе, что и проблему трансляции. Исходная программа писалась на проблемно-ориентированном языке без учета особенностей технических средств, на которых она будет реализована. Такое игнорирование особенностей

средств реализации налагает принципиальное ограничение на качество транслируемых программ, поскольку это качество в сильной мере зависит от выявления таких свойств транслируемой программы, которые бы наилучшим образом согласовывались с особенностями средств реализации. Но, как известно, возможности формального выявления указанных свойств в общем случае являются неразрешимой проблемой.

По этой же причине формальные методы не могут в общем случае обеспечить высокое качество распараллеливания. Однако если низкое качество трансляции программ может быть в некоторых случаях допустимым (например, для разовых задач с небольшим объемом счета), то низкое качество распараллеливания просто недопустимо, так как нужда в параллельной реализации программ и возникает при необходимости повышения эффективности вычислений.

2.3. Созданная мощная технологическая база позволяет выпускать ЭВМ с оперативными памятями больших объёмов. Традиционно сложившаяся в начальный период тенденция экономии объема памяти за счет увеличения объема вычислений приводит к недоиспользованию возможностей оперативных памятей современных ЭВМ и, как следствие этого, к увеличению времени вычислений.

2.4. Модель машины фон Неймана, лежащая в основе традиционных языков программирования, оказалась неудобной для представления алгоритмов ряда классов задач и в то же время достаточно далека от конкретных реализаций многопроцессорных систем [I].

2.5. Отсутствие жесткой регламентации привело к возникновению огромного множества принципиально не отличающихся друг от друга языков программирования. На программистском фольклоре это нашло отражение в виде девиза: "Каждый уважающий себя программист должен создать свой язык".Поскольку "уважающих себя" (и соответственно "не уважающих" других) оказалось достаточно много,то в программировании возникла ситуация вавилонского многоязычия, крайне затруднившая широкое использование уже имеющихся программ, на написание и отладку которых был затрачен огромный труд и дорогостоящее машинное время.Не говоря уже о том, во что обошлись сами разработки многочисленных языков и соответствующих трансляторов, а также их освоение и эксплуатация.

2.6. Стоимость математического обеспечения стремительно растет и уже начинает превышать стоимость оборудования. Если сохранится существующая тенденция, то стоимость математического обеспечения станет в недалеком будущем сопоставимой с годовым бюдже-

том страны. Наряду с ростом стоимости программного продукта все отчетливее ощущается тенденция сокращения сроков его реального использования. Быстрое моральное старение, а следовательно, обесценивание огромного труда программистов часто связано не с возникновением новых, более эффективных методов вычислений и не с возникновением новых принципов построения технических средств, а порождается отсутствием преемственности у вновь создаваемых версий языков, операционных систем и т.п. с уже имеющимися. Большой вклад в удорожание и в моральное старение математического обеспечения вносит и существующая в этой области разноголосица, о которой шла речь выше.

3. Итак, мы установили, что традиционная парадигма оказалась бессильной разрешить противоречие между требованием сделать общение человека с ЭВМ доступным неквалифицированному пользователю и требованием высокой эффективности использования возможностей технических средств.

Попытаемся установить, какие же свойства должны быть присущи новой парадигме для того, чтобы на ее основе стало реальным преодоление указанного противоречия.

Рассмотрим прежде всего методологические особенности парадигмы, характеризующие ту научную базу, на которой строится математическое обеспечение. Уделим основное внимание следующим трем группам свойств:

- свойствам изначальных элементов - атомов, из которых строится программное обеспечение;

- свойствам методов синтеза функциональных структур, т.е. методов построения из атомов заданной функциональной системы.

- свойствам методов синтеза ресурсных структур, т.е. методов распределения имеющихся технических средств (ресурсов) между элементами функциональной системы (атомами и их объединениями - блоками), адаптации этих элементов к заданным параметрам технических средств для достижения высокоэффективной организации вычислений и решения собственно задачи синтеза, заключающейся в нахождении оптимальных вариантов объединения атомов в блоки, тех, в свою очередь, в блоки более высокого уровня и т.д. до объединения их в единую систему.

3.1. Свойства атомов.

3.1.1. Функциональные свойства атомов являются более или менее традиционными. Каждому из атомов поставлена в соответствие вы-

полняемая им функция, перечень задаваемых параметров решаемой задачи и допустимый диапазон их изменения, а также перечень тех атомов, с которыми допускается функциональное взаимодействие.

3.I.2. Менее традиционными являются ресурсные свойства атомов. Эти свойства включают в себя перечисление ресурсов,необходимых для работы данного атома, допустимые диапазоны изменения каждого из ресурсов, а также перечень атомов,с которыми данный атом может вступать в ресурсное взаимодействие того или иного типа. В согласии с [2] среди ресурсных схем взаимодействия атомов выделим: параллельную схему, когда атомы работают и потребляют ресурсы одновременно, и последовательные, когда атомы работают поочередно. Последовательные схемы, в свою очередь,подразделяются на схемы без маневра ресурсами (ресурсы предварительно распределяются между атомами и далее не перераспределяются) и схемы с маневром ресурсами, когда данный вид ресурса (например, процессоры или оперативная память) поочередно используются каждым из атомов.И наконец, смешанные схемы, когда подобный маневр допускается только для некоторых видов ресурсов (например, только для процессоров или только для памятей).

3.I.3. Одно из важных свойств атомов - преемственность ресурсных свойств, которое состоит в том,что при объединении дачной совокупности атомов одновременно и в функциональную,и в ресурсную структуру получается функциональный блок, обладающий теми же ресурсными свойствами, что и атомы. Иначе говоря, в атомы заложено (так сказать, генетически) свойство комплексирования в себеподобные блоки, способные выполнять более сложные функции, т.е. эти блоки также могут объединяться в более сложные блоки, обладающие теми же ресурсными свойствами.

3.I.4. Предполагается,что время счета каждого из атомов связано с количеством предоставляемых ресурсов строго убывающей (по каждому из видов ресурсов) функциональной зависимостью в пределах всей области определения.

3.I.5. Среди ресурсов, на которые в первую очередь должны ориентироваться атомы, выделим два основных: количество процессоров и объем оперативной памяти.

Как показали исследования, проделанные нами еще в 60-х годах [3], простым и эффективным методом разделения процесса вычислений на заданное число параллельных ветвей (по числу процессоров) является распараллеливание по циклам [4]. Этот метод предполагает

наличие у алгоритмов циклов, повторения которых могут выполнять - ся независимо друг от друга. Например, циклы над массивами данных. В наиболее простом и часто встречающемся случае, когда число повторений цикла L фиксировано и совпадает с размером массива, распределение вычислений между l процессорами сводится к разделению массива на l частей (l = 1 ÷ L) . При этом загрузка процессоров будет достаточно равномерной как при $l \ll L$, так и при L , кратных l (или мало отличающихся от кратных).

Использование указанного свойства циклов ограничивает набор атомов теми, которые реализуют циклы над массивами данных с относительно большим числом повторений. Иными словами, у атомов должен существовать определенный порог сложности.

3.I.6. Зависимость времени счета атома от предоставленного ему объема оперативной памяти является предметом дальнейших исследований. На сегодняшний день подобные зависимости изучены лишь для отдельных классов задач. Но и этого оказалось достаточным для того, чтобы убедиться в перспективности подобных исследований. Так, одна из первых работ в этой области, возникшая в связи с разработкой и опытной эксплуатацией однородной вычислительной системы "Минск-222", показала, что при объединении нескольких ЭВМ в систему эффект от увеличения суммарного объема оперативных памятей нередко сопоставим или даже превышает эффект от увеличения числа процессоров [5]. Для одних задач этот эффект достигался благодаря уменьшению времени обмена с внешними памятями (по этому поводу см. также [6]), для других благодаря применению табличных методов счета, для третьих из-за более удобного (развернутого) представления данных.

Заметим также, что не обязательно весь диапазон допустимого изменения объема оперативной памяти должен покрываться одним и тем же алгоритмом. Вполне допустимо применение и набора алгоритмов, эквивалентных функционально, но отличающихся диапазоном использования ресурсов. Это же замечание может касаться и использования другого вида ресурсов - числа процессоров.

3.2. Методы синтеза функциональных структур нацелены на автоматическое комплексирование из имеющегося набора атомов (и блоков, созданных ранее) нового блока, способного решить задачу, описан - ную пользователем. По своему характеру эти методы сходны с авто - матическим доказательством теорем и машинными методами поискового конструирования [7]. Как показывает опыт, существуют классы задач,

для которых подобные методы синтеза могут успешно работать [6]. Однако более правильно сказать, что разработки этих методов находятся еще в начальной стадии.

3.3. Методы синтеза ресурсных структур также находятся в начальной стадии своего развития. За последнее время удалось создать основы теории идеально организованных (гармоничных) иерархических систем [2], которая позволяет в принципе решить эту проблему для определенного вида функциональной зависимости между эффективностью функционирования (в данном случае - временем счета) атома и количеством отведенных ему ресурсов. В согласии с этой теорией указанная зависимость должна выражаться с помощью степенной функции, т.е. должна иметь тот же вид, который обычно встречается в выражениях, описывающих физические законы. В вопросе, насколько это требование приемлемо для широкого класса программных объектов, полной ясности пока нет. Имеются многочисленные примеры, подтверждающие степенной (или квазистепенной) характер зависимости времени счета от числа процессоров [3]. Имеются также отдельные подтверждающие примеры и для зависимости времени счета алгоритма от предоставленного для его реализации объема оперативной памяти [5]. Все это говорит о том, что имеются основания для веры в успешность создания достаточно представительной библиотеки атомов с требуемыми свойствами и синтеза из них функциональных блоков с желаемыми свойствами.

Литература

1. СВИРИДЕНКО Д.И. О природе программирования. -В кн.: Математическое обеспечение ВС из микро-ЭВМ (Вычислительные системы, вып. 96). Новосибирск, 1983, с. 51-74.

2. КОСАРЕВ Ю.Г. О математической модели гармоничных систем. - Там же, с. 3-28.

3. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные универсальные вычислительные системы высокой производительности. -Новосибирск: Наука, 1966. - 308 с.

4. КОСАРЕВ Ю.Г. Распараллеливание по циклам. -В кн.: Вычислительные системы, вып. 24. Новосибирск, 1967, с. 3-20.

5. КОСАРЕВ Ю.Г. Примеры использования таблиц для сокращения времени счета. -В кн.: Вычислительные системы, вып. 30. Новосибирск, 1968, с. 46-54.

6. ПЕТРОВА Л.Т. Алгоритмическое построение задач оптимальной обработки составных файлов. -В кн.: Проблемы нелинейного программирования (Оптимизация, вып. 22(39)), Новосибирск, 1978, с. 103-114.

7. АВТОМАТИЗАЦИЯ поискового конструирования (искусственный интеллект в машинном проектировании)/ А.И.Половинкин, Н.К.Бобков, Г.Я.Буш и др.; под ред. А.И.Половинкина. -М.: Радио и связь,1981. - 344 с.

8. ЧУКАНОВА Н.А. Индуктивная система программирования для массового пользователя. -В кн.: Р-технология программирования: Тез. докл. I Всесоюзн. конф. 4.П. Опыт применения. Киев, ИК АН УССР, 1983, с. 185-188.

Поступила в ред.-изд. отд.
21 декабря 1983 год